DNSOP Working Group
Internet-Draft
Updates: RFC 2845, RFC 2931 (if approved) (if approved)
Intended status: Standards Track
Expires: May 3, 2018

R. Bellis
ISC
P. van Dijk
R. Gacogne
PowerDNS

October 30, 2017

DNS X-Proxied-For
draft-bellis-dnsop-xpf-03

Abstract

It is becoming more commonplace to install front end proxy devices in front of DNS servers to provide (for example) load balancing or to perform transport layer conversions.

This document defines a meta resource record that allows a DNS server to receive information about the client's original transport protocol parameters when supplied by trusted proxies.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

It is becoming more commonplace to install front end proxy devices in front of DNS servers [RFC1035] to provide load balancing or to perform transport layer conversions (e.g. to add DNS over TLS [RFC7858] to a DNS server that lacks native support).

This has the unfortunate side effect of hiding the clients' source IP addresses from the server, making it harder to employ server-side technologies that rely on knowing those addresses (e.g. ACLs, DNS Response Rate Limiting, etc).

This document defines the XPF meta resource record (RR) that allows a DNS server to receive information about the client's original transport protocol parameters when supplied by trusted proxies.

Whilst in some circumstances it would be possible to re-use the Client Subnet EDNS Option [RFC7871] to carry a subset of this information, a new RR is defined to allow both this feature and the Client Subnet Option to co-exist in the same packet.

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.

The XPF RR is analogous to the HTTP "X-Forwarded-For" header, but in DNS the term "forwarder" is usually understood to describe a network component that sits on the outbound query path of a resolver.

Instead we use the term "proxy", which in this document means a network component that sits on the inbound query path in front of a recursive or authoritative DNS server, receiving DNS queries from clients and dispatching them to local servers.

3. Description

The XPF RR contains the entire 6-tuple (IP version, Layer 4 protocol, source address, destination address, source port and destination port) of the packet received from the client by the proxy.

The presence of the source address supports use of ACLs based on the client’s IP address.

The source port allows for ACLs to support Carrier Grade NAT whereby different end-users might share a single IP address.

The destination address supports scenarios where the server behaviour depends upon the packet destination (e.g. BIND view’s "match-destinations" option)

The protocol and destination port fields allow server behaviour to vary depending on whether DNS over TLS [RFC7858] or DNS over DTLS [RFC8094] are in use.

3.1. Client Handling

Stub resolvers, client-side proxy devices, and recursive resolvers MUST NOT add the XPF RR to DNS requests.

3.2. Request Handling

The rules in this section apply to processing of the XPF RR whether by a proxy device or a DNS server.

If this RR is received from a non-white-listed client the server MUST return a REFUSED response.

If a server finds this RR anywhere other than in the Additional Section of a request it MUST return a REFUSED response.

If the value of the RR's IP version field is not understood by the server it MUST return a REFUSED response.

If the length of the IP addresses contained in the RR are not consistent with that expected for the given IP version then the server MUST return a FORMERR response.

Servers MUST NOT send this RR in DNS responses.

3.3. Proxy Handling

For each request received, proxies MUST generate an XPF RR containing the 6-tuple representing the client's Layer 3 and Layer 4 headers and append it to the Additional Section of the request (updating the ARCOUNT field accordingly) before sending it to the intended DNS server.

If a valid XPF RR is received from a white-listed client the original XPF RR MUST be preserved instead.

3.4. Server Handling

When this RR is received from a white-listed client the DNS server SHOULD use the transport information contained therein in preference to the packet's own transport information for any data processing logic (e.g. ACLs) that would otherwise depend on the latter.

3.5. Wire Format

The XPF RR is formatted like any standard RR, but none of the fields except RDLENGTH and RDATA have any meaning in this specification. All multi-octet fields are transmitted in network order (i.e. big-endian).

The required values of the RR header fields are as follows:

NAME: MUST contain a single 0 octet (i.e. the root domain).

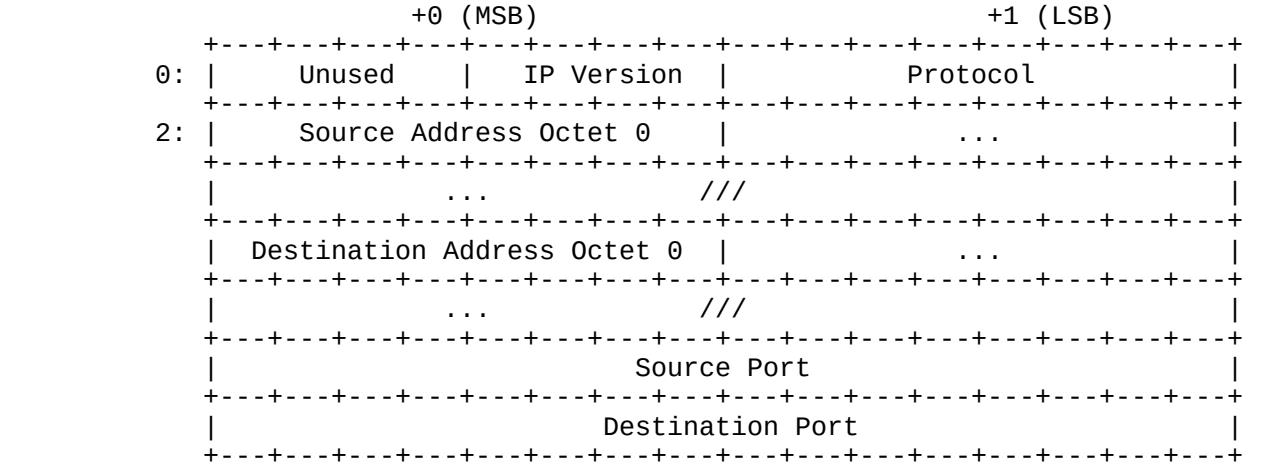
TYPE: MUST contain TBD1 (XPF).

CLASS: MUST contain 1 (IN).

TTL: MUST contain 0 (zero).

RDLENGTH: specifies the length in octets of the RDATA field.

The RDATA of the XPF RR is as follows:



Unused: Currently reserved. These bits MUST be zero unless redefined in a subsequent specification.

IP Version: The IP protocol version number used by the client, as defined in the IANA IP Version Number Registry [IANA-IP]. Implementations MUST support IPv4 (4) and IPv6 (6).

Protocol: The Layer 4 protocol number (e.g. UDP or TCP) as defined in the IANA Protocol Number Registry [IANA-PROTO].

Source Address: The source IP address of the client.

Destination Address: The destination IP address of the request, i.e. the IP address of the proxy on which the request was received.

Source Port: The source port used by the client.

Destination Port: The destination port of the request.

The length of the Source Address and Destination Address fields will be variable depending on the IP Version used by the client.

3.6. Presentation Format

XPF is a meta RR that cannot appear in master format zone files, but a standardised presentation format is defined here for use by debugging utilities that might need to display the contents of an XPF RR.

The Unused bits and the IP Version field are treated as a single octet and presented as an unsigned decimal integer with range 0 .. 255.

The Protocol field is presented as an unsigned decimal integer with range 0 .. 255.

The Source and Destination Address fields are presented either as IPv4 or IPv6 addresses according to the IP Version field. In the case of IPv6 the recommendations from [RFC5952] SHOULD be followed.

The Source and Destination Port fields are presented as unsigned decimal integers with range 0 .. 65535.

3.7. Signed DNS Requests

Any XPF RRs found in a packet MUST be ignored for the purposes of calculating or verifying any signatures used for Secret Key Transaction Authentication for DNS [RFC2845] or DNS Request and Transaction Signatures (SIG(0)) [RFC2931].

Typically it is expected that proxies will append the XPF RR to the packet after any existing TSIG or SIG(0) RRs, and that servers will remove the XPF RR from the packet prior to verification of the original signature, with the ARCOUNT field updated as appropriate.

If either TSIG or SIG(0) are configured between the proxy and server then any XPF RRs MUST be ignored when the proxy calculates the packet signature.

4. Security Considerations

If the white-list of trusted proxies is implemented as a list of IP addresses, the server administrator MUST have the ability to selectively disable this feature for any transport where there is a possibility of the proxy’s source address being spoofed.

This does not mean to imply that use over UDP is impossible - if for example the network architecture keeps all proxy-to-server traffic on a dedicated network and clients have no direct access to the servers then the proxies’ source addresses can be considered unspoofable.

5. Privacy Considerations

Used incorrectly, this RR could expose internal network information, however it is not intended for use on proxy / forwarder devices that sit on the client-side of a DNS request.

This specification is only intended for use on server-side proxy devices that are under the same administrative control as the DNS servers themselves. As such there is no change in the scope within which any private information might be shared.

Use other than as described above would be contrary to the principles of [RFC6973].

6. IANA Considerations

<< a copy of the RFC 6895 IANA RR TYPE application template will appear here >>

7. Acknowledgements

Mark Andrews, Robert Edmonds, Duane Wessels

8. References

8.1. Normative References

[IANA-IP] IANA, "IANA IP Version Registry", n.d., <http://www.iana.org/assignments/version-numbers/>.

[IANA-PROTO] IANA, "IANA Protocol Number Registry", n.d., <http://www.iana.org/assignments/protocol-numbers/>.

[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>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

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

[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>.

8.2. Informative References

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[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>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[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>.

Authors' Addresses

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

Phone: +1 650 423 1200
Email: ray@isc.org

Peter van Dijk
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: peter.van.dijk@powerdns.com

Remi Gacogne
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: remi.gacogne@powerdns.com

DNSOP Working Group
Internet-Draft
Updates: RFC 2845, RFC 2931 (if approved) (if approved)
Intended status: Standards Track
Expires: September 6, 2018

R. Bellis
ISC
P. van Dijk
R. Gacogne
PowerDNS

March 05, 2018

DNS X-Proxied-For
draft-bellis-dnsop-xpf-04

Abstract

It is becoming more commonplace to install front end proxy devices in front of DNS servers to provide (for example) load balancing or to perform transport layer conversions.

This document defines a meta resource record that allows a DNS server to receive information about the client's original transport protocol parameters when supplied by trusted proxies.

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 September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

It is becoming more commonplace to install front end proxy devices in front of DNS servers [RFC1035] to provide load balancing or to perform transport layer conversions (e.g. to add DNS over TLS [RFC7858] to a DNS server that lacks native support).

This has the unfortunate side effect of hiding the clients' source IP addresses from the server, making it harder to employ server-side technologies that rely on knowing those addresses (e.g. ACLs, DNS Response Rate Limiting, etc).

This document defines the XPF meta resource record (RR) that allows a DNS server to receive information about the client's original transport protocol parameters when supplied by trusted proxies.

Whilst in some circumstances it would be possible to re-use the Client Subnet EDNS Option [RFC7871] to carry a subset of this

information, a new RR is defined to allow both this feature and the Client Subnet Option to co-exist in the same packet.

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.

The XPF RR is analogous to the HTTP "X-Forwarded-For" header, but in DNS the term "forwarder" is usually understood to describe a network component that sits on the outbound query path of a resolver.

Instead we use the term "proxy", which in this document means a network component that sits on the inbound query path in front of a recursive or authoritative DNS server, receiving DNS queries from clients and dispatching them to local servers.

3. Description

The XPF RR contains the entire 6-tuple (IP version, Layer 4 protocol, source address, destination address, source port and destination port) of the packet received from the client by the proxy.

The presence of the source address supports use of ACLs based on the client’s IP address.

The source port allows for ACLs to support Carrier Grade NAT whereby different end-users might share a single IP address.

The destination address supports scenarios where the server behaviour depends upon the packet destination (e.g. BIND view’s "match-destinations" option)

The protocol and destination port fields allow server behaviour to vary depending on whether DNS over TLS [RFC7858] or DNS over DTLS [RFC8094] are in use.

3.1. Client Handling

Stub resolvers, client-side proxy devices, and recursive resolvers MUST NOT add the XPF RR to DNS requests.

3.2. Request Handling

The rules in this section apply to processing of the XPF RR whether by a proxy device or a DNS server.

If this RR is received from a non-white-listed client the server MUST return a REFUSED response.

If a server finds this RR anywhere other than in the Additional Section of a request it MUST return a REFUSED response.

If the value of the RR's IP version field is not understood by the server it MUST return a REFUSED response.

If the length of the IP addresses contained in the RR are not consistent with that expected for the given IP version then the server MUST return a FORMERR response.

Servers MUST NOT send this RR in DNS responses.

3.3. Proxy Handling

For each request received, proxies MUST generate an XPF RR containing the 6-tuple representing the client's Layer 3 and Layer 4 headers and append it to the Additional Section of the request (updating the ARCOUNT field accordingly) before sending it to the intended DNS server.

If a valid XPF RR is received from a white-listed client the original XPF RR MUST be preserved instead.

3.4. Server Handling

When this RR is received from a white-listed client the DNS server SHOULD use the transport information contained therein in preference to the packet's own transport information for any data processing logic (e.g. ACLs) that would otherwise depend on the latter.

3.5. Wire Format

The XPF RR is formatted like any standard RR, but none of the fields except RDLENGTH and RDATA have any meaning in this specification. All multi-octet fields are transmitted in network order (i.e. big-endian).

The required values of the RR header fields are as follows:

NAME: MUST contain a single 0 octet (i.e. the root domain).

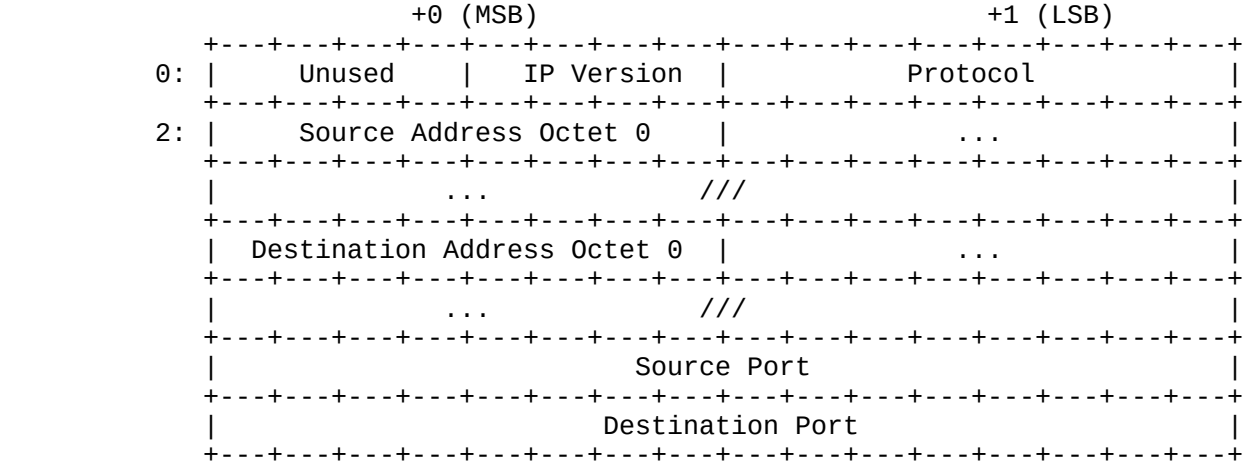
TYPE: MUST contain TBD1 (XPF).

CLASS: MUST contain 1 (IN).

TTL: MUST contain 0 (zero).

RDLENGTH: specifies the length in octets of the RDATA field.

The RDATA of the XPF RR is as follows:



Unused: Currently reserved. These bits MUST be zero unless redefined in a subsequent specification.

IP Version: The IP protocol version number used by the client, as defined in the IANA IP Version Number Registry [IANA-IP]. Implementations MUST support IPv4 (4) and IPv6 (6).

Protocol: The Layer 4 protocol number (e.g. UDP or TCP) as defined in the IANA Protocol Number Registry [IANA-PROTO].

Source Address: The source IP address of the client.

Destination Address: The destination IP address of the request, i.e. the IP address of the proxy on which the request was received.

Source Port: The source port used by the client.

Destination Port: The destination port of the request.

The length of the Source Address and Destination Address fields will be variable depending on the IP Version used by the client.

3.6. Presentation Format

XPF is a meta RR that cannot appear in master format zone files, but a standardised presentation format is defined here for use by debugging utilities that might need to display the contents of an XPF RR.

The Unused bits and the IP Version field are treated as a single octet and presented as an unsigned decimal integer with range 0 .. 255.

The Protocol field is presented as an unsigned decimal integer with range 0 .. 255.

The Source and Destination Address fields are presented either as IPv4 or IPv6 addresses according to the IP Version field. In the case of IPv6 the recommendations from [RFC5952] SHOULD be followed.

The Source and Destination Port fields are presented as unsigned decimal integers with range 0 .. 65535.

3.7. Signed DNS Requests

Any XPF RRs found in a packet MUST be ignored for the purposes of calculating or verifying any signatures used for Secret Key Transaction Authentication for DNS [RFC2845] or DNS Request and Transaction Signatures (SIG(0)) [RFC2931].

Typically it is expected that proxies will append the XPF RR to the packet after any existing TSIG or SIG(0) RRs, and that servers will remove the XPF RR from the packet prior to verification of the original signature, with the ARCOUNT field updated as appropriate.

If either TSIG or SIG(0) are configured between the proxy and server then any XPF RRs MUST be ignored when the proxy calculates the packet signature.

4. Security Considerations

If the white-list of trusted proxies is implemented as a list of IP addresses, the server administrator MUST have the ability to selectively disable this feature for any transport where there is a possibility of the proxy’s source address being spoofed.

This does not mean to imply that use over UDP is impossible - if for example the network architecture keeps all proxy-to-server traffic on a dedicated network and clients have no direct access to the servers then the proxies’ source addresses can be considered unspoofable.

5. Implementation status

[RFC Editor Note: Please remove this entire section prior to publication as an RFC.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. dnsdist

Support for adding an XPF RR to proxied packets is provided in the git version of dnsdist. The code point is configurable.

5.2. PowerDNS Recursor

Support for extracting the XPF RR from received packets (when coming from a trusted source) is available in the git version of the PowerDNS Recursor. The code point is configurable.

5.3. Wireshark

Support for dissecting XPF RRs is present in Wireshark 2.5.0, using a temporary code point of 65422.

6. Privacy Considerations

Used incorrectly, this RR could expose internal network information, however it is not intended for use on proxy / forwarder devices that sit on the client-side of a DNS request.

This specification is only intended for use on server-side proxy devices that are under the same administrative control as the DNS servers themselves. As such there is no change in the scope within which any private information might be shared.

Use other than as described above would be contrary to the principles of [RFC6973].

7. IANA Considerations

<< a copy of the RFC 6895 IANA RR TYPE application template will appear here >>

8. Acknowledgements

Mark Andrews, Robert Edmonds, Duane Wessels

9. References

9.1. Normative References

[IANA-IP] IANA, "IANA IP Version Registry", n.d., <http://www.iana.org/assignments/version-numbers/>.

[IANA-PROTO] IANA, "IANA Protocol Number Registry", n.d., <http://www.iana.org/assignments/protocol-numbers/>.

[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>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

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

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

9.2. Informative References

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[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>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[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>.

Authors' Addresses

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

Phone: +1 650 423 1200
Email: ray@isc.org

Peter van Dijk
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: peter.van.dijk@powerdns.com

Remi Gacogne
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: remi.gacogne@powerdns.com

Internet Engineering Task Force
Internet-Draft
Obsoletes: 2845, 4635 (if approved)
Intended status: Standards Track
Expires: May 3, 2018

F. Dupont, Ed.
S. Morris
ISC

October 30, 2017

Secret Key Transaction Authentication for DNS (TSIG)
draft-dupont-dnsop-rfc2845bis-00

Abstract

This protocol allows for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates as coming from an approved client, or to authenticate responses as coming from an approved recursive name server.

No provision has been made here for distributing the shared secrets: it is expected that a network administrator will statically configure name servers and clients using some out of band mechanism such as sneaker-net until a secure automated mechanism for key distribution is available.

This document includes revised original TSIG specifications (RFC2845) and the extension for HMAC-SHA (RFC4635).

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 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

In 2017, security problems in two nameservers strictly following [RFC2845] and [RFC4635] (i.e., TSIG and HMAC-SHA extension) specifications were discovered. The implementations were fixed but, to avoid similar problems in the future, the two documents were updated and merged, producing these revised specifications for TSIG.

The Domain Name System (DNS) [RFC1034], [RFC1035] is a replicated hierarchical distributed database system that provides information fundamental to Internet operations, such as name <=> address translation and mail handling information.

This document specifies use of a message authentication code (MAC), either HMAC-MD5 or HMAC-SHA (keyed hash functions), to provide an efficient means of point-to-point authentication and integrity checking for transactions.

The second area where the secret key based MACs specified in this document can be used is to authenticate DNS update requests as well as transaction responses, providing a lightweight alternative to the protocol described by [RFC3007].

A further use of this mechanism is to protect zone transfers. In this case the data covered would be the whole zone transfer including any glue records sent. The protocol described by DNSSEC does not protect glue records and unsigned records unless SIG(0) (transaction signature) is used.

The authentication mechanism proposed in this document uses shared secret keys to establish a trust relationship between two entities. Such keys must be protected in a fashion similar to private keys, lest a third party masquerade as one of the intended parties (forge MACs). There is an urgent need to provide simple and efficient authentication between clients and local servers and this proposal addresses that need. This proposal is unsuitable for general server to server authentication for servers which speak with many other servers, since key management would become unwieldy with the number of shared keys going up quadratically. But it is suitable for many resolvers on hosts that only talk to a few recursive servers.

A server acting as an indirect caching resolver -- a "forwarder" in common usage -- might use transaction-based authentication when communicating with its small number of preconfigured "upstream" servers. Other uses of DNS secret key authentication and possible systems for automatic secret key distribution may be proposed in separate future documents.

Note that use of TSIG presumes prior agreement between the resolver and server involved as to the algorithm and key to be used.

Since the publication of first version of this document ([RFC2845]) a mechanism based on asymmetric signatures using the SIG RR was specified (SIG(0) [RFC2931]) when this document uses symmetric authentication codes calculated by HMAC [RFC2104] using strong hash functions.

2. Key words

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. New Assigned Numbers

RRTYPE = TSIG (250)
ERROR = 0..15 (a DNS RCODE)
ERROR = 16 (BADSIG)
ERROR = 17 (BADKEY)

ERROR = 18 (BADTIME)
ERROR = 22 (BADTRUNC)

4. TSIG RR Format

4.1. TSIG RR Type

To provide secret key authentication, we use a new RR type whose mnemonic is TSIG and whose type code is 250. TSIG is a meta-RR and MUST NOT be cached. TSIG RRs are used for authentication between DNS entities that have established a shared secret key. TSIG RRs are dynamically computed to cover a particular DNS transaction and are not DNS RRs in the usual sense.

4.2. TSIG Calculation

As the TSIG RRs are related to one DNS request/response, there is no value in storing or retransmitting them, thus the TSIG RR is discarded once it has been used to authenticate a DNS message. All multi-octet integers in the TSIG record are sent in network byte order (see [RFC1035] 2.3.2).

4.3. TSIG Record Format

NAME The name of the key used in domain name syntax. The name should reflect the names of the hosts and uniquely identify the key among a set of keys these two hosts may share at any given time. If hosts A.site.example and B.example.net share a key, possibilities for the key name include <id>.A.site.example, <id>.B.example.net, and <id>.A.site.example.B.example.net. It should be possible for more than one key to be in simultaneous use among a set of interacting hosts. The name only needs to be meaningful to the communicating hosts but a meaningful mnemonic name as above is strongly recommended.

The name may be used as a local index to the key involved and it is recommended that it be globally unique. Where a key is just shared between two hosts, its name actually only need only be meaningful to them but it is recommended that the key name be mnemonic and incorporate the resolver and server host names in that order.

TYPE TSIG (250: Transaction SIGnature)

CLASS ANY

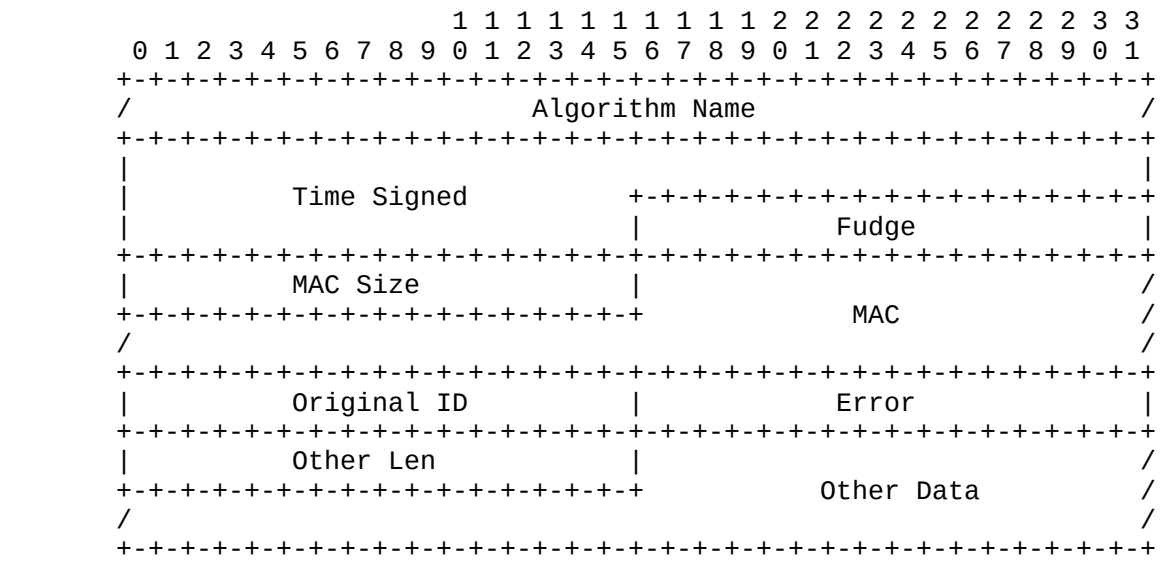
TTL 0

RdLen (variable)

RDATA

4.3.1. TSIG RDATA Wire Format

The RDATA for a TSIG RR consists of an octet stream Algorithm Name field, a uint48_t Time Signed field, a uint16_t Fudge field, a uint16_t MAC Size field, a octet stream MAC field, a uint16_t Original ID, a uint16_t Error field, a uint16_t Other Len field and an octet stream of Other Data.



4.3.1.1. The Algorithm Name Field

The Algorithm Name field identifies the TSIG algorithm name in the domain name syntax.

4.3.1.2. The Time Signed Field

The Time Signed field specifies seconds since 1970-01-01 UTC.

4.3.1.3. The Fudge Field

The Fudge field specifies allowed time difference in seconds permitted in the Time Signed field.

4.3.1.4. The MAC Size Field

The MAC Size field specifies the length of MAC field in octets. Truncation is indicated by a MAC size less than the HMAC size.

4.3.1.5. The MAC Field

The MAC field contents are defined by the used Algorithm.

4.3.1.6. The Error field

The Error field contains the Expanded RCODE covering TSIG processing.

4.3.1.7. The Other Len Field

The Other Len field specifies the length of Other Data in octets.

4.3.1.8. The Other Data Field

The Other Data field is empty unless Error == BADTIME.

4.4. Example

NAME HOST.EXAMPLE.

TYPE TSIG

CLASS ANY

TTL 0

RdLen As appropriate

RDATA

Field Name	Contents
Algorithm Name	SAMPLE-ALG.EXAMPLE.
Time Signed	853804800
Fudge	300
MAC Size	As appropriate
MAC	As appropriate
Original ID	As appropriate
Error	0 (NOERROR)
Other Len	0
Other Data	Empty

5. Protocol Operation

5.1. Effects of adding TSIG to outgoing message

Once the outgoing message has been constructed, the keyed message digest operation can be performed. The resulting message digest will then be stored in a TSIG which is appended to the additional data section (the ARCOUNT is incremented to reflect this). If the TSIG record cannot be added without causing the message to be truncated, the server MUST alter the response so that a TSIG can be included. This response consists of only the question and a TSIG record, and has the TC bit set and RCODE 0 (NOERROR). The client SHOULD at this point retry the request using TCP (per [RFC1035] 4.2.2).

5.2. TSIG processing on incoming messages

If an incoming message contains a TSIG record, it MUST be the last record in the additional section. Multiple TSIG records are not allowed. If a TSIG record is present in any other position, the packet is dropped and a response with RCODE 1 (FORMERR) MUST be returned. Upon receipt of a message with a correctly placed TSIG RR, the TSIG RR is copied to a safe location, removed from the DNS Message, and decremented out of the DNS message header's ARCOUNT. At this point the keyed message digest operation is performed: until this operation concludes that the signature is valid, the signature MUST be considered to be invalid. If the algorithm name or key name is unknown to the recipient, or if the message digests do not match, the whole DNS message MUST be discarded. If the message is a query, a response with RCODE 9 (NOTAUTH) MUST be sent back to the originator with TSIG ERROR 17 (BADKEY) or TSIG ERROR 16 (BADSIG). If no key is available to sign this message it MUST be sent unsigned (MAC size == 0 and empty MAC). A message to the system operations log SHOULD be generated, to warn the operations staff of a possible security incident in progress. Care should be taken to ensure that logging of this type of event does not open the system to a denial of service attack.

5.3. Time values used in TSIG calculations

The data digested includes the two timer values in the TSIG header in order to defend against replay attacks. If this were not done, an attacker could replay old messages but update the "Time Signed" and "Fudge" fields to make the message look new. This data is named "TSIG Timers", and for the purpose of digest calculation they are invoked in their "on the wire" format, in the following order: first Time Signed, then Fudge. For example:

Field Name	Value	Wire Format	Meaning
Time Signed	853804800	00 00 32 e4 07 00	Tue Jan 21 00:00:00 1997
Fudge	300	01 2C	5 minutes

5.4. TSIG Variables and Coverage

When generating or verifying the contents of a TSIG record, the following data are digested, in network byte order or wire format, as appropriate:

5.4.1. DNS Message

A whole and complete DNS message in wire format, before the TSIG RR has been added to the additional data section and before the DNS Message Header's ARCOUNT field has been incremented to contain the TSIG RR. If the message ID differs from the original message ID, the original message ID is substituted for the message ID. This could happen when forwarding a dynamic update request, for example.

5.4.2. TSIG Variables

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	(Always ANY in the current specification)
TSIG RR	TTL	(Always 0 in the current specification)
TSIG RDATA	Algorithm Name	in canonical wire format
TSIG RDATA	Time Signed	in network byte order
TSIG RDATA	Fudge	in network byte order
TSIG RDATA	Error	in network byte order
TSIG RDATA	Other Len	in network byte order
TSIG RDATA	Other Data	exactly as transmitted

The RR RDLEN and RDATA MAC Length are not included in the hash since they are not guaranteed to be knowable before the MAC is generated.

The Original ID field is not included in this section, as it has already been substituted for the message ID in the DNS header and hashed.

For each label type, there must be a defined "Canonical wire format" that specifies how to express a label in an unambiguous way. For label type 00, this is defined in [RFC4034], for label type 01, this is defined in [RFC6891]. The use of label types other than 00 and 01 is not defined for this specification.

5.4.3. Request MAC

When generating the MAC to be included in a response, the validated request MAC MUST be included in the digest. If the request MAC failed to validate, an unsigned error message MUST be returned instead. (Section 6.3).

The request's MAC is digested in wire format, including the following fields:

Field	Type	Description
MAC Length	uint16_t	in network byte order
MAC Data	octet stream	exactly as transmitted

5.5. Padding

Digested components are fed into the hashing function as a continuous octet stream with no interfield padding.

6. Protocol Details

6.1. TSIG generation on requests

Client performs the message digest operation and appends a TSIG record to the additional data section and transmits the request to the server. The client MUST store the message digest from the request while awaiting an answer. The digest components for a request are:

DNS Message (request)
TSIG Variables (request)

Note that some older name servers will not accept requests with a nonempty additional data section. Clients SHOULD only attempt signed transactions with servers who are known to support TSIG and share some secret key with the client -- so, this is not a problem in practice.

6.2. TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST NOT generate a signed response to an unsigned request or a request that fails validation. The digest components are:

Request MAC
DNS Message (response)

TSIG Variables (response)

6.3. TSIG on TSIG Error returns

When a server detects an error relating to the key or MAC, the server SHOULD send back an unsigned error message (MAC size == 0 and empty MAC). If an error is detected relating to the TSIG validity period or the MAC is too short for the local policy, the server SHOULD send back a signed error message. The digest components are:

Request MAC (if the request MAC validated)
DNS Message (response)
TSIG Variables (response)

The reason that the request is not included in this digest in some cases is to make it possible for the client to verify the error. If the error is not a TSIG error the response MUST be generated as specified in Section 6.2.

6.4. TSIG on TCP connection

A DNS TCP session can include multiple DNS envelopes. This is, for example, commonly used by zone transfer. Using TSIG on such a connection can protect the connection from hijacking and provide data integrity. The TSIG MUST be included on the first and last DNS envelopes. It can be optionally placed on any intermediary envelopes. It is expensive to include it on every envelopes, but it MUST be placed on at least every 100’th envelope. The first envelope is processed as a standard answer, and subsequent messages have the following digest components:

Prior Digest (running)
DNS Messages (any unsigned messages since the last TSIG)
TSIG Timers (current message)

This allows the client to rapidly detect when the session has been altered; at which point it can close the connection and retry. If a client TSIG verification fails, the client MUST close the connection. If the client does not receive TSIG records frequently enough (as specified above) it SHOULD assume the connection has been hijacked and it SHOULD close the connection. The client SHOULD treat this the same way as they would any other interrupted transfer (although the exact behavior is not specified).

6.5. Server TSIG checks

Upon receipt of a message, server will check if there is a TSIG RR. If one exists, the server is REQUIRED to return a TSIG RR in the response. The server MUST perform the following checks in the following order, check Key, check MAC, check Time values, check Truncation policy.

6.5.1. Key check and error handling

If a non-forwarding server does not recognize the key used by the client, the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 17 (BADKEY). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error.

6.5.2. Specifying Truncation

When space is at a premium and the strength of the full length of an HMAC is not needed, it is reasonable to truncate the HMAC and use the truncated value for authentication. HMAC SHA-1 truncated to 96 bits is an option available in several IETF protocols, including IPsec and TLS.

Processing of a truncated MAC follows these rules

1. If "MAC size" field is greater than HMAC output length:

   This case MUST NOT be generated and, if received, MUST cause the packet to be dropped and RCODE 1 (FORMERR) to be returned.

2. If "MAC size" field equals HMAC output length:

   The entire output HMAC output is present and used.

3. "MAC size" field is less than HMAC output length but greater than that specified in case 4, below:

   This is sent when the signer has truncated the HMAC output to an allowable length, as described in [RFC2104], taking initial octets and discarding trailing octets. TSIG truncation can only be to an integral number of octets. On receipt of a packet with truncation thus indicated, the locally calculated MAC is similarly truncated and only the truncated values are compared for authentication. The request MAC used when calculating the TSIG MAC for a reply is the truncated request MAC.

4. "MAC size" field is less than the larger of 10 (octets) and half the length of the hash function in use:

With the exception of certain TSIG error messages described in Section 6.3, where it is permitted that the MAC size be zero, this case MUST NOT be generated and, if received, MUST cause the packet to be dropped and RCODE 1 (FORMERR) to be returned.

6.5.3. MAC check and error handling

If a TSIG fails to verify, the server MUST generate an error response as specified in Section 6.3 with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error.

6.5.4. Time check and error handling

If the server time is outside the time interval specified by the request (which is: Time Signed, plus/minus Fudge), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 18 (BADTIME). The server SHOULD also cache the most recent time signed value in a message generated by a key, and SHOULD return BADTIME if a message received later has an earlier time signed value. A response indicating a BADTIME error MUST be signed by the same key as the request. It MUST include the client's current time in the time signed field, the server's current time (a uint48_t) in the other data field, and 6 in the other data length field. This is done so that the client can verify a message with a BADTIME error without the verification failing due to another BADTIME error. The data signed is specified in Section 6.3. The server SHOULD log the error.

6.5.5. Truncation check and error handling

If a TSIG is received with truncation that is permitted under Section 6.5.2 above but the MAC is too short for the local policy in force, an RCODE 9 (NOTAUTH) and TSIG ERROR 22 (BADTRUNC) MUST be returned. The server SHOULD log the error.

6.6. Client processing of answer

When a client receives a response from a server and expects to see a TSIG, it first checks if the TSIG RR is present in the response. Otherwise, the response is treated as having a format error and discarded. The client then extracts the TSIG, adjusts the ARCOUNT, and calculates the keyed digest in the same way as the server, applying the same rules to decide if truncated MAC is valid. If the TSIG does not validate, that response MUST be discarded, unless the RCODE is 9 (NOTAUTH), in which case the client SHOULD attempt to verify the response as if it were a TSIG Error response, as specified in Section 6.3. A message containing an unsigned TSIG record or a TSIG record which fails verification SHOULD NOT be considered an

acceptable response; the client SHOULD log an error and continue to wait for a signed response until the request times out.

6.6.1. Key error handling

If an RCODE on a response is 9 (NOTAUTH), and the response TSIG validates, and the TSIG key is different from the key used on the request, then this is a Key error. The client MAY retry the request using the key specified by the server. This should never occur, as a server MUST NOT sign a response with a different key than signed the request.

6.6.2. MAC error handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC error, and client MAY retry the request with a new request ID but it would be better to try a different shared key if one is available. Clients SHOULD keep track of how many MAC errors are associated with each key. Clients SHOULD log this event.

6.6.3. Time error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 18 (BADTIME), or the current time does not fall in the range specified in the TSIG record, then this is a Time error. This is an indication that the client and server clocks are not synchronized. In this case the client SHOULD log the event. DNS resolvers MUST NOT adjust any clocks in the client based on BADTIME errors, but the server’s time in the other data field SHOULD be logged.

6.6.4. Truncation error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 22 (BADTRUNC) the this is a Truncation error. The client MAY retry with lesser truncation up to the full HMAC output (no truncation), using the truncation used in the response as a hint for what the server policy allowed (Section 8). Clients SHOULD log this event.

6.7. Special considerations for forwarding servers

A server acting as a forwarding server of a DNS message SHOULD check for the existence of a TSIG record. If the name on the TSIG is not of a secret that the server shares with the originator the server MUST forward the message unchanged including the TSIG. If the name of the TSIG is of a key this server shares with the originator, it MUST process the TSIG. If the TSIG passes all checks, the forwarding server MUST, if possible, include a TSIG of his own, to the destination or the next forwarder. If no transaction security is

available to the destination and the response has the AD flag (see [RFC4035]), the forwarder MUST unset the AD flag before adding the TSIG to the answer.

7. Algorithms and Identifiers

The only message digest algorithm specified in the first version of these specifications [RFC2845] was "HMAC-MD5" (see [RFC1321], [RFC2104]). The "HMAC-MD5" algorithm is mandatory to implement for interoperability.

The use of SHA-1 [FIPS180-4], [RFC3174], (which is a 160-bit hash as compared to the 128 bits for MD5), and additional hash algorithms in the SHA family [FIPS180-4], [RFC3874], [RFC6234] with 224, 256, 384, and 512 bits may be preferred in some cases. This is because increasingly successful cryptanalytic attacks are being made on the shorter hashes.

Use of TSIG between a DNS resolver and server is by mutual agreement. That agreement can include the support of additional algorithms and criteria as to which algorithms and truncations are acceptable, subject to the restriction and guidelines in Section 6.5.2 above. Key agreement can be by the TKEY mechanism [RFC2930] or some other mutually agreeable method.

The current HMAC-MD5.SIG-ALG.REG.INT and gss-tsig identifiers are included in the table below for convenience. Implementations that support TSIG MUST also implement HMAC SHA1 and HMAC SHA256 and MAY implement gss-tsig and the other algorithms listed below.

Requirement	Name
Mandatory	HMAC-MD5.SIG-ALG.REG.INT
Optional	gss-tsig
Mandatory	hmac-sha1
Optional	hmac-sha224
Mandatory	hmac-sha256
Optional	hmac-sha384
Optional	hmac-sha512

SHA-1 truncated to 96 bits (12 octets) SHOULD be implemented.

8. TSIG Truncation Policy

Use of TSIG is by mutual agreement between a resolver and server. Implicit in such an "agreement" are criteria as to acceptable keys and algorithms and, with the extensions in this document, truncations. Note that it is common for implementations to bind the

TSIG secret key or keys that may be in place at a resolver and server to particular algorithms. Thus, such implementations only permit the use of an algorithm if there is an associated key in place. Receipt of an unknown, unimplemented, or disabled algorithm typically results in a BADKEY error.

Local policies MAY require the rejection of TSIGs, even though they use an algorithm for which implementation is mandatory.

When a local policy permits acceptance of a TSIG with a particular algorithm and a particular non-zero amount of truncation, it SHOULD also permit the use of that algorithm with lesser truncation (a longer MAC) up to the full HMAC output.

Regardless of a lower acceptable truncated MAC length specified by local policy, a reply SHOULD be sent with a MAC at least as long as that in the corresponding request. Note if the request specified a MAC length longer than the HMAC output it will be rejected by processing rules Section 6.5.2 case 1.

Implementations permitting multiple acceptable algorithms and/or truncations SHOULD permit this list to be ordered by presumed strength and SHOULD allow different truncations for the same algorithm to be treated as separate entities in this list. When so implemented, policies SHOULD accept a presumed stronger algorithm and truncation than the minimum strength required by the policy.

9. Shared Secrets

Secret keys are very sensitive information and all available steps should be taken to protect them on every host on which they are stored. Generally such hosts need to be physically protected. If they are multi-user machines, great care should be taken that unprivileged users have no access to keying material. Resolvers often run unprivileged, which means all users of a host would be able to see whatever configuration data is used by the resolver.

A name server usually runs privileged, which means its configuration data need not be visible to all users of the host. For this reason, a host that implements transaction-based authentication should probably be configured with a "stub resolver" and a local caching and forwarding name server. This presents a special problem for [RFC2136] which otherwise depends on clients to communicate only with a zone’s authoritative name servers.

Use of strong random shared secrets is essential to the security of TSIG. See [RFC4086] for a discussion of this issue. The secret

SHOULD be at least as long as the keyed message digest, i.e., 16 bytes for HMAC-MD5 or 20 bytes for HMAC-SHA1.

10. IANA Considerations

IANA maintains a registry of algorithm names to be used as "Algorithm Names" as defined in Section 4.3. Algorithm names are text strings encoded using the syntax of a domain name. There is no structure required other than names for different algorithms must be unique when compared as DNS names, i.e., comparison is case insensitive. Previous specifications [RFC2845] and [RFC4635] defined values for HMAC MD5 and SHA. IANA has also registered "gss-tsig" as an identifier for TSIG authentication where the cryptographic operations are delegated to the Generic Security Service (GSS) [RFC3645].

New algorithms are assigned using the IETF Consensus policy defined in [RFC8126]. The algorithm name HMAC-MD5.SIG-ALG.REG.INT looks like a fully-qualified domain name for historical reasons; other algorithm names are simple (i.e., single-component) names.

IANA maintains a registry of "TSIG Error values" to be used for "Error" values as defined in Section 4.3. Initial values should be those defined in Section 3. New TSIG error codes for the TSIG error field are assigned using the IETF Consensus policy defined in [RFC8126].

11. Security Considerations

The approach specified here is computationally much less expensive than the signatures specified in DNSSEC. As long as the shared secret key is not compromised, strong authentication is provided for the last hop from a local name server to the user resolver.

Secret keys should be changed periodically. If the client host has been compromised, the server should suspend the use of all secrets known to that client. If possible, secrets should be stored in encrypted form. Secrets should never be transmitted in the clear over any network. This document does not address the issue on how to distribute secrets. Secrets should never be shared by more than two entities.

This mechanism does not authenticate source data, only its transmission between two parties who share some secret. The original source data can come from a compromised zone master or can be corrupted during transit from an authentic zone master to some "caching forwarder." However, if the server is faithfully performing the full DNSSEC security checks, then only security checked data will be available to the client.

A fudge value that is too large may leave the server open to replay attacks. A fudge value that is too small may cause failures if machines are not time synchronized or there are unexpected network delays. The recommended value in most situation is 300 seconds.

For all of the message authentication code algorithms listed in this document, those producing longer values are believed to be stronger; however, while there have been some arguments that mild truncation can strengthen a MAC by reducing the information available to an attacker, excessive truncation clearly weakens authentication by reducing the number of bits an attacker has to try to break the authentication by brute force [RFC2104].

Significant progress has been made recently in cryptanalysis of hash functions of the types used here, all of which ultimately derive from the design of MD4. While the results so far should not effect HMAC, the stronger SHA-1 and SHA-256 algorithms are being made mandatory due to caution. Note that today SHA-3 [FIPS202] is available as an alternative to SHA-2 using a very different design.

See also the Security Considerations section of [RFC2104] from which the limits on truncation in this RFC were taken.

11.1. Issue fixed in this document

To bind an answer with its corresponding request the MAC of the answer is computed using the MAC request. Unfortunately original specifications [RFC2845] failed to clearly require the MAC request to be successfully validated.

This document proposes the principle that the MAC must be considered to be invalid until it was validated. This leads to the requirement that only a validated request MAC is included in a signed answer. Or with other words when the request MAC was not validated the answer must be unsigned with a BADKEY or BADSIG TSIG error.

11.2. Why not DNSSEC?

This section from the original document [RFC2845] analyzes DNSSEC in order to justify the introduction of TSIG.

DNS has recently been extended by DNSSEC ([RFC4033], [RFC4034] and [RFC4035]) to provide for data origin authentication, and public key distribution, all based on public key cryptography and public key based digital signatures. To be practical, this form of security generally requires extensive local caching of keys and tracing of authentication through multiple keys and signatures to a pre-trusted locally configured key.

One difficulty with the DNSSEC scheme is that common DNS implementations include simple "stub" resolvers which do not have caches. Such resolvers typically rely on a caching DNS server on another host. It is impractical for these stub resolvers to perform general DNSSEC authentication and they would naturally depend on their caching DNS server to perform such services for them. To do so securely requires secure communication of queries and responses. DNSSEC provides public key transaction signatures to support this, but such signatures are very expensive computationally to generate. In general, these require the same complex public key logic that is impractical for stubs.

A second area where use of straight DNSSEC public key based mechanisms may be impractical is authenticating dynamic update [RFC2136] requests. DNSSEC provides for request signatures but with DNSSEC they, like transaction signatures, require computationally expensive public key cryptography and complex authentication logic. Secure Domain Name System Dynamic Update ([RFC3007]) describes how different keys are used in dynamically updated zones.

12. References

12.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC4635] Eastlake 3rd, D., "HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers", RFC 4635, DOI 10.17487/RFC4635, August 2006, <https://www.rfc-editor.org/info/rfc4635>.

[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>.

12.2. Informative References

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard", FIPS PUB 202, August 2015.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <https://www.rfc-editor.org/info/rfc2930>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/rfc3007>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, DOI 10.17487/RFC3645, October 2003, <https://www.rfc-editor.org/info/rfc3645>.

[RFC3874] Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, DOI 10.17487/RFC3874, September 2004, <https://www.rfc-editor.org/info/rfc3874>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[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>.

[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>.

Appendix A. Acknowledgments

This document just consolidates and updates the earlier documents by the authors of [RFC2845] (Paul Vixie, Olafur Gudmundsson, Donald E. Eastlake 3rd and Brian Wellington) and [RFC4635] (Donald E. Eastlake 3rd). It would not be possible without their original work.

The security problem addressed by this document was reported by Clement Berthaux from Synacktiv.

Note for the RFC Editor (to be removed before publication): the first ’e’ in Clement is a fact a small ’e’ with acute, unicode code U+00E9. I do not know if xml2rfc supports non ASCII characters so I prefer to not experiment with it. BTW I am French too too so I can help if you have questions like correct spelling...

Peter van Dijk, Benno Overeinder, Willem Toroop, Ondrej Sury, Mukund Sivaraman and Ralph Dolmans participated in the discussions that prompted this document.

Appendix B. Change History

draft-dupont-dnsop-rfc2845bis-00

[RFC4635] was merged.

Authors of original documents were moved to Acknowledgments (Appendix A).

Section 2 was updated to [RFC8174] style.

Spit references into normative and informative references and updated them.

Added a text explaining why this document was written in the Abstract and at the beginning of the introduction.

Clarified the layout of TSIG RDATA.

Moved the text about using DNSSEC from the Introduction to the end of Security Considerations.

Added the security clarifications:

1. Emphasized that MAC is invalid until it is successfully validated.

2. Added requirement that a request MAC that has not been successfully validated MUST NOT be included into a response.

3. Added requirement that a request that has not been validated to the MUST NOT generate a signed response.

4. Added note about MAC too short for the local policy to the Section 6.3.

5. Changed the order of server checks and swapped corresponding sections.

6. Removed the truncation size limit "also case" as it does not apply and added confusion.

7. Relocated the error provision for TSIG truncation to the new Section 6.5.5. Moved from RCODE 22 to RCODE 9 and TSIG ERROR 22, i.e., aligned with other TSIG error cases.

8. Added Section 6.6.4 about truncation error handling by clients.

9. Removed the limit to HMAC output in replies as a request which specified a MAC length longer than the HMAC output is invalid according the the first processing rule in Section 6.5.2.

10. Promoted the requirement that a secret length should be at least as long as the keyed message digest to a SHOULD [RFC2119] key word.

11. Added a short text to explain the security issue.

Authors' Addresses

Francis Dupont (editor)
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States

Email: Francis.Dupont@fdupont.fr

Stephen Morris
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States

Email: stephen@isc.org
URI: http://www.isc.org

Internet Engineering Task Force
Internet-Draft
Obsoletes: 2845, 4635 (if approved)
Intended status: Standards Track
Expires: September 6, 2018

F. Dupont, Ed.
S. Morris
ISC

March 5, 2018

Secret Key Transaction Authentication for DNS (TSIG)
draft-dupont-dnsop-rfc2845bis-01

Abstract

This protocol allows for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates as coming from an approved client, or to authenticate responses as coming from an approved name server.

No provision has been made here for distributing the shared secrets: it is expected that a network administrator will statically configure name servers and clients using some out of band mechanism.

This document includes revised original TSIG specifications (RFC2845) and its extension for HMAC-SHA (RFC4635).

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 September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

In 2017, security problems in two nameservers strictly following [RFC2845] and [RFC4635] (i.e., TSIG and its HMAC-SHA extension) specifications were discovered. The implementations were fixed but, to avoid similar problems in the future, the two documents were updated and merged, producing these revised specifications for TSIG.

The Domain Name System (DNS) [RFC1034], [RFC1035] is a replicated hierarchical distributed database system that provides information fundamental to Internet operations, such as name <=> address translation and mail handling information.

This document specifies use of a message authentication code (MAC), either HMAC-MD5 or HMAC-SHA (keyed hash functions), to provide an efficient means of point-to-point authentication and integrity checking for transactions.

The second area where the secret key based MACs specified in this document can be used is to authenticate DNS update requests as well as transaction responses, providing a lightweight alternative to the protocol described by [RFC3007].

A further use of this mechanism is to protect zone transfers. In this case the data covered would be the whole zone transfer including any glue records sent. The protocol described by DNSSEC does not

protect glue records and unsigned records unless SIG(0) (transaction signature) is used.

The authentication mechanism proposed in this document uses shared secret keys to establish a trust relationship between two entities. Such keys must be protected in a fashion similar to private keys, lest a third party masquerade as one of the intended parties (by forging the MAC). There is an urgent need to provide simple and efficient authentication between clients and local servers and this proposal addresses that need. The proposal is unsuitable for general server to server authentication for servers which speak with many other servers, since key management would become unwieldy with the number of shared keys going up quadratically. But it is suitable for many resolvers on hosts that only talk to a few recursive servers.

A server acting as an indirect caching resolver -- a "forwarder" in common usage -- might use transaction-based authentication when communicating with its small number of preconfigured "upstream" servers. Other uses of DNS secret key authentication and possible systems for automatic secret key distribution may be proposed in separate future documents.

Note that use of TSIG presumes prior agreement between the two parties involved (e.g., resolver and server) as to the algorithm and key to be used.

Since the publication of first version of this document ([RFC2845]) a mechanism based on asymmetric signatures using the SIG RR was specified (SIG(0) [RFC2931]) whereas this document uses symmetric authentication codes calculated by HMAC [RFC2104] using strong hash functions.

2. Key words

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. New Assigned Numbers

RRTYPE = TSIG (250)
ERROR = 0..15 (a DNS RCODE)
ERROR = 16 (BADSIG)
ERROR = 17 (BADKEY)
ERROR = 18 (BADTIME)
ERROR = 22 (BADTRUNC)

4. TSIG RR Format

4.1. TSIG RR Type

To provide secret key authentication, we use a new RR type whose mnemonic is TSIG and whose type code is 250. TSIG is a meta-RR and MUST NOT be cached. TSIG RRs are used for authentication between DNS entities that have established a shared secret key. TSIG RRs are dynamically computed to cover a particular DNS transaction and are not DNS RRs in the usual sense.

4.2. TSIG Calculation

As the TSIG RRs are related to one DNS request/response, there is no value in storing or retransmitting them, thus the TSIG RR is discarded once it has been used to authenticate a DNS message. Recommendations concerning the message digest agorithm can be found in Section 7. All multi-octet integers in the TSIG record are sent in network byte order (see [RFC1035] 2.3.2).

4.3. TSIG Record Format

NAME The name of the key used in domain name syntax. The name should reflect the names of the hosts and uniquely identify the key among a set of keys these two hosts may share at any given time. If hosts A.site.example and B.example.net share a key, possibilities for the key name include <id>.A.site.example, <id>.B.example.net, and <id>.A.site.example.B.example.net. It should be possible for more than one key to be in simultaneous use among a set of interacting hosts. The name only needs to be meaningful to the communicating hosts but a meaningful mnemonic name as above is strongly recommended.

The name may be used as a local index to the key involved and it is recommended that it be globally unique. Where a key is just shared between two hosts, its name actually need only be meaningful to them but it is recommended that the key name be mnemonic and incorporate the resolver and server host names in that order.

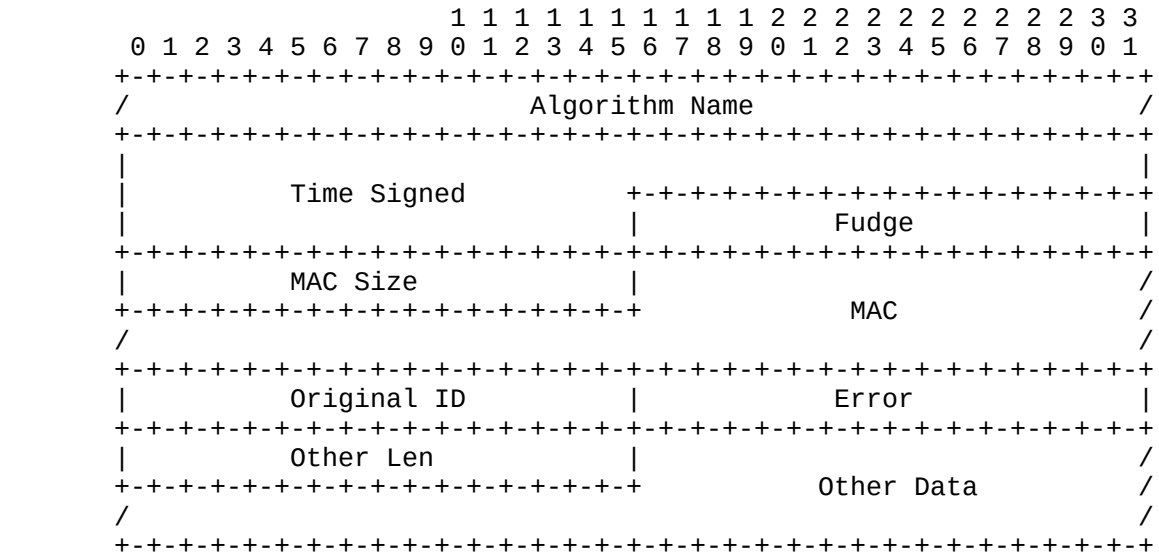
TYPE TSIG (250: Transaction SIGnature)

CLASS ANY

TTL 0

RdLen (variable)

RDATA The RDATA for a TSIG RR consists of an octet stream Algorithm Name field, a uint48_t Time Signed field, a uint16_t Fudge field, a uint16_t MAC Size field, a octet stream MAC field, a uint16_t Original ID, a uint16_t Error field, a uint16_t Other Len field and an octet stream of Other Data.



The contents of the RDATA fields are:

* Algorithm Name - identifies the TSIG algorithm name in the domain name syntax.

* Time Signed - the The Time Signed field specifies seconds since 00:00 on 1970-01-01 UTC.

* Fudge - specifies allowed time difference in seconds permitted in the Time Signed field.

* MAC Size - the MAC Size field specifies the length of MAC field in octets. Truncation is indicated by a MAC size less than the HMAC size.

* MAC - the contents of the MAC field are defined by the TSIG algorithm used.

* Error - contains the expanded RCODE covering TSIG processing.

* Other Len - specifies the length of the "Other Data" field in octets.

* Other Data - this field will be empty unless the content of the Error field is BADTIME, in which case it will contain the server's current time (see Section 6.5.4).

4.4. Example

NAME HOST.EXAMPLE.

TYPE TSIG

CLASS ANY

TTL 0

RdLen As appropriate

RDATA

Field Name	Contents
Algorithm Name	SAMPLE-ALG.EXAMPLE.
Time Signed	853804800
Fudge	300
MAC Size	As appropriate
MAC	As appropriate
Original ID	As appropriate
Error	0 (NOERROR)
Other Len	0
Other Data	Empty

5. Protocol Operation

5.1. Effects of adding TSIG to outgoing message

Once the outgoing message has been constructed, the HMAC computation can be performed. The resulting MAC will then be stored in a TSIG which is appended to the additional data section (the ARCOUNT is incremented to reflect this). If the TSIG record cannot be added without causing the message to be truncated, the server MUST alter the response so that a TSIG can be included. This response consists of only the question and a TSIG record, and has the TC bit set and RCODE 0 (NOERROR). The client SHOULD at this point retry the request using TCP (per [RFC1035] 4.2.2).

5.2. TSIG processing on incoming messages

If an incoming message contains a TSIG record, it MUST be the last record in the additional section. Multiple TSIG records are not allowed. If a TSIG record is present in any other position, the DNS message is dropped and a response with RCODE 1 (FORMERR) MUST be returned. Upon receipt of a message with a correctly placed TSIG RR, the TSIG RR is copied to a safe location, removed from the DNS Message, and decremented out of the DNS message header's ARCOUNT. At this point the HMAC computation is performed: until this operation concludes that the signature is valid, the signature MUST be considered to be invalid.

If the algorithm name or key name is unknown to the recipient, or if the MACs do not match, the whole DNS message MUST be discarded. If the message is a query, a response with RCODE 9 (NOTAUTH) MUST be sent back to the originator with TSIG ERROR 17 (BADKEY) or TSIG ERROR 16 (BADSIG). If no key is available to sign this message it MUST be sent unsigned (MAC size == 0 and empty MAC). A message to the system operations log SHOULD be generated, to warn the operations staff of a possible security incident in progress. Care should be taken to ensure that logging of this type of event does not open the system to a denial of service attack.

5.3. Time values used in TSIG calculations

The data digested includes the two timer values in the TSIG header in order to defend against replay attacks. If this were not done, an attacker could replay old messages but update the "Time Signed" and "Fudge" fields to make the message look new. This data is named "TSIG Timers", and for the purpose of MAC calculation they are invoked in their "on the wire" format, in the following order: first Time Signed, then Fudge. For example:

Field Name	Value	Wire Format	Meaning
Time Signed	853804800	00 00 32 e4 07 00	Tue Jan 21 00:00:00 1997
Fudge	300	01 2C	5 minutes

5.4. TSIG Variables and Coverage

When generating or verifying the contents of a TSIG record, the following data are passed as input to MAC computation, in network byte order or wire format, as appropriate:

5.4.1. DNS Message

A whole and complete DNS message in wire format, before the TSIG RR has been added to the additional data section and before the DNS Message Header's ARCOUNT field has been incremented to contain the TSIG RR. If the message ID differs from the original message ID, the original message ID is substituted for the message ID. This could happen when forwarding a dynamic update request, for example.

5.4.2. TSIG Variables

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	(Always ANY in the current specification)
TSIG RR	TTL	(Always 0 in the current specification)
TSIG RDATA	Algorithm Name	in canonical wire format
TSIG RDATA	Time Signed	in network byte order
TSIG RDATA	Fudge	in network byte order
TSIG RDATA	Error	in network byte order
TSIG RDATA	Other Len	in network byte order
TSIG RDATA	Other Data	exactly as transmitted

The RR RDLEN and RDATA MAC Length are not included in the input to MAC computation since they are not guaranteed to be knowable before the MAC is generated.

The Original ID field is not included in this section, as it has already been substituted for the message ID in the DNS header and hashed.

For each label type, there must be a defined "Canonical wire format" that specifies how to express a label in an unambiguous way. For label type 00, this is defined in [RFC4034], for label type 01, this is defined in [RFC6891]. The use of label types other than 00 and 01 is not defined for this specification.

5.4.3. Request MAC

When generating the MAC to be included in a response, the validated request MAC MUST be included in the MAC computation. If the request MAC failed to validate, an unsigned error message MUST be returned instead. (Section 6.3).

The request's MAC is digested in wire format, including the following fields:

Field	Type	Description
MAC Length	uint16_t	in network byte order
MAC Data	octet stream	exactly as transmitted

5.5. Padding

Digested components (i.e., inputs to HMAC computation) are fed into the hashing function as a continuous octet stream with no interfield padding.

6. Protocol Details

6.1. TSIG generation on requests

Client performs the HMAC computation and appends a TSIG record to the additional data section and transmits the request to the server. The client MUST store the MAC from the request while awaiting an answer. The digest components for a request are:

```
DNS Message (request)
TSIG Variables (request)
```

Note that some older name servers will not accept requests with a nonempty additional data section. Clients SHOULD only attempt signed transactions with servers who are known to support TSIG and share some secret key with the client -- so, this is not a problem in practice.

6.2. TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST NOT generate a signed response to an unsigned request or a request that fails validation. The digest components are:

```
Request MAC
DNS Message (response)
TSIG Variables (response)
```

6.3. TSIG on TSIG Error returns

When a server detects an error relating to the key or MAC, the server SHOULD send back an unsigned error message (MAC size == 0 and empty MAC). If an error is detected relating to the TSIG validity period or the MAC is too short for the local policy, the server SHOULD send back a signed error message. The digest components are:

```
Request MAC (if the request MAC validated)
DNS Message (response)
TSIG Variables (response)
```

The reason that the request is not included in this MAC in some cases is to make it possible for the client to verify the error. If the error is not a TSIG error the response MUST be generated as specified in Section 6.2.

6.4. TSIG on zone tranfer over a TCP connection

A zone transfer over a DNS TCP session can include multiple DNS messages. Using TSIG on such a connection can protect the connection from hijacking and provide data integrity. The TSIG MUST be included on the first and last DNS messages, and for new implementations SHOULD be placed on all intermediary messages. For backward compatibility the client which receives DNS messages and verifies TSIG MUST accept up to 99 intermediary messages without a TSIG. The first envelope is processed as a standard answer, and subsequent messages have the following digest components:

```
Prior MAC (running)
DNS Messages (any unsigned messages since the last TSIG)
TSIG Timers (current message)
```

This allows the client to rapidly detect when the session has been altered; at which point it can close the connection and retry. If a client TSIG verification fails, the client MUST close the connection. If the client does not receive TSIG records frequently enough (as specified above) it SHOULD assume the connection has been hijacked and it SHOULD close the connection. The client SHOULD treat this the same way as they would any other interrupted transfer (although the exact behavior is not specified).

6.5. Server TSIG checks

Upon receipt of a message, server will check if there is a TSIG RR. If one exists, the server is REQUIRED to return a TSIG RR in the response. The server MUST perform the following checks in the following order, check Key, check MAC, check Time values, check Truncation policy.

6.5.1. Key check and error handling

If a non-forwarding server does not recognize the key used by the client, the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 17 (BADKEY). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error.

6.5.2. Specifying Truncation

When space is at a premium and the strength of the full length of an HMAC is not needed, it is reasonable to truncate the HMAC and use the truncated value for authentication. HMAC SHA-1 truncated to 96 bits is an option available in several IETF protocols, including IPsec and TLS.

Processing of a truncated MAC follows these rules

1. If "MAC size" field is greater than HMAC output length:

   This case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.

2. If "MAC size" field equals HMAC output length:

   The entire output HMAC output is present and used.

3. "MAC size" field is less than HMAC output length but greater than that specified in case 4, below:

   This is sent when the signer has truncated the HMAC output to an allowable length, as described in [RFC2104], taking initial octets and discarding trailing octets. TSIG truncation can only be to an integral number of octets. On receipt of a DNS message with truncation thus indicated, the locally calculated MAC is similarly truncated and only the truncated values are compared for authentication. The request MAC used when calculating the TSIG MAC for a reply is the truncated request MAC.

4. "MAC size" field is less than the larger of 10 (octets) and half the length of the hash function in use:

   With the exception of certain TSIG error messages described in Section 6.3, where it is permitted that the MAC size be zero, this case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.

6.5.3. MAC check and error handling

If a TSIG fails to verify, the server MUST generate an error response as specified in Section 6.3 with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error.

6.5.4. Time check and error handling

If the server time is outside the time interval specified by the request (which is: Time Signed, plus/minus Fudge), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 18 (BADTIME). The server SHOULD also cache the most recent time signed value in a message generated by a key, and SHOULD return BADTIME if a message received later has an earlier time signed value. A response indicating a BADTIME error MUST be signed by the same key as the request. It MUST include the client's current time in the time signed field, the server's current time (a uint48_t) in the other data field, and 6 in the other data length field. This is done so that the client can verify a message with a BADTIME error without the verification failing due to another BADTIME error. The data signed is specified in Section 6.3. The server SHOULD log the error.

6.5.5. Truncation check and error handling

If a TSIG is received with truncation that is permitted under Section 6.5.2 above but the MAC is too short for the local policy in force, an RCODE 9 (NOTAUTH) and TSIG ERROR 22 (BADTRUNC) MUST be returned. The server SHOULD log the error.

6.6. Client processing of answer

When a client receives a response from a server and expects to see a TSIG, it first checks if the TSIG RR is present in the response. Otherwise, the response is treated as having a format error and discarded. The client then extracts the TSIG, adjusts the ARCOUNT, and calculates the MAC in the same way as the server, applying the same rules to decide if truncated MAC is valid. If the TSIG does not validate, that response MUST be discarded, unless the RCODE is 9 (NOTAUTH), in which case the client SHOULD attempt to verify the response as if it were a TSIG Error response, as specified in Section 6.3. A message containing an unsigned TSIG record or a TSIG record which fails verification SHOULD NOT be considered an acceptable response; the client SHOULD log an error and continue to wait for a signed response until the request times out.

6.6.1. Key error handling

If an RCODE on a response is 9 (NOTAUTH), and the response TSIG validates, and the TSIG key is different from the key used on the request, then this is a Key error. The client MAY retry the request using the key specified by the server. This should never occur, as a server MUST NOT sign a response with a different key than signed the request.

6.6.2. MAC error handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC error, and client MAY retry the request with a new request ID but it would be better to try a different shared key if one is available. Clients SHOULD keep track of how many MAC errors are associated with each key. Clients SHOULD log this event.

6.6.3. Time error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 18 (BADTIME), or the current time does not fall in the range specified in the TSIG record, then this is a Time error. This is an indication that the client and server clocks are not synchronized. In this case the client SHOULD log the event. DNS resolvers MUST NOT adjust any clocks in the client based on BADTIME errors, but the server's time in the other data field SHOULD be logged.

6.6.4. Truncation error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 22 (BADTRUNC) the this is a Truncation error. The client MAY retry with lesser truncation up to the full HMAC output (no truncation), using the truncation used in the response as a hint for what the server policy allowed (Section 8). Clients SHOULD log this event.

6.7. Special considerations for forwarding servers

A server acting as a forwarding server of a DNS message SHOULD check for the existence of a TSIG record. If the name on the TSIG is not of a secret that the server shares with the originator the server MUST forward the message unchanged including the TSIG. If the name of the TSIG is of a key this server shares with the originator, it MUST process the TSIG. If the TSIG passes all checks, the forwarding server MUST, if possible, include a TSIG of his own, to the destination or the next forwarder. If no transaction security is available to the destination and the response has the AD flag (see [RFC4035]), the forwarder MUST unset the AD flag before adding the TSIG to the answer.

7. Algorithms and Identifiers

The only message digest algorithm specified in the first version of these specifications [RFC2845] was "HMAC-MD5" (see [RFC1321], [RFC2104]). The "HMAC-MD5" algorithm is mandatory to implement for interoperability.

The use of SHA-1 [FIPS180-4], [RFC3174], (which is a 160-bit hash as compared to the 128 bits for MD5), and additional hash algorithms in the SHA family [FIPS180-4], [RFC3874], [RFC6234] with 224, 256, 384, and 512 bits may be preferred in some cases. This is because increasingly successful cryptanalytic attacks are being made on the shorter hashes.

Use of TSIG between two DNS agents is by mutual agreement. That agreement can include the support of additional algorithms and criteria as to which algorithms and truncations are acceptable, subject to the restriction and guidelines in Section 6.5.2 above. Key agreement can be by the TKEY mechanism [RFC2930] or some other mutually agreeable method.

The current HMAC-MD5.SIG-ALG.REG.INT and gss-tsig identifiers are included in the table below for convenience. Implementations that support TSIG MUST also implement HMAC SHA1 and HMAC SHA256 and MAY implement gss-tsig and the other algorithms listed below.

Requirement	Name
Mandatory	HMAC-MD5.SIG-ALG.REG.INT
Optional	gss-tsig
Mandatory	hmac-sha1
Optional	hmac-sha224
Mandatory	hmac-sha256
Optional	hmac-sha384
Optional	hmac-sha512

SHA-1 truncated to 96 bits (12 octets) SHOULD be implemented.

8. TSIG Truncation Policy

Use of TSIG is by mutual agreement between two DNS agents, e.g., a resolver and server. Implicit in such an "agreement" are criteria as to acceptable keys and algorithms and, with the extensions in this document, truncations. Note that it is common for implementations to bind the TSIG secret key or keys that may be in place at two parties to particular algorithms. Thus, such implementations only permit the use of an algorithm if there is an associated key in place. Receipt of an unknown, unimplemented, or disabled algorithm typically results in a BADKEY error.

Local policies MAY require the rejection of TSIGs, even though they use an algorithm for which implementation is mandatory.

When a local policy permits acceptance of a TSIG with a particular algorithm and a particular non-zero amount of truncation, it SHOULD

also permit the use of that algorithm with lesser truncation (a longer MAC) up to the full HMAC output.

Regardless of a lower acceptable truncated MAC length specified by local policy, a reply SHOULD be sent with a MAC at least as long as that in the corresponding request. Note if the request specified a MAC length longer than the HMAC output it will be rejected by processing rules Section 6.5.2 case 1.

Implementations permitting multiple acceptable algorithms and/or truncations SHOULD permit this list to be ordered by presumed strength and SHOULD allow different truncations for the same algorithm to be treated as separate entities in this list. When so implemented, policies SHOULD accept a presumed stronger algorithm and truncation than the minimum strength required by the policy.

9. Shared Secrets

Secret keys are very sensitive information and all available steps should be taken to protect them on every host on which they are stored. Generally such hosts need to be physically protected. If they are multi-user machines, great care should be taken that unprivileged users have no access to keying material. Resolvers often run unprivileged, which means all users of a host would be able to see whatever configuration data is used by the resolver.

A name server usually runs privileged, which means its configuration data need not be visible to all users of the host. For this reason, a host that implements transaction-based authentication should probably be configured with a "stub resolver" and a local caching and forwarding name server. This presents a special problem for [RFC2136] which otherwise depends on clients to communicate only with a zone’s authoritative name servers.

Use of strong random shared secrets is essential to the security of TSIG. See [RFC4086] for a discussion of this issue. The secret SHOULD be at least as long as the HMAC output, i.e., 16 bytes for HMAC-MD5 or 20 bytes for HMAC-SHA1.

10. IANA Considerations

IANA maintains a registry of algorithm names to be used as "Algorithm Names" as defined in Section 4.3. Algorithm names are text strings encoded using the syntax of a domain name. There is no structure required other than names for different algorithms must be unique when compared as DNS names, i.e., comparison is case insensitive. Previous specifications [RFC2845] and [RFC4635] defined values for HMAC MD5 and SHA. IANA has also registered "gss-tsig" as an

identifier for TSIG authentication where the cryptographic operations are delegated to the Generic Security Service (GSS) [RFC3645].

New algorithms are assigned using the IETF Consensus policy defined in [RFC8126]. The algorithm name HMAC-MD5.SIG-ALG.REG.INT looks like a fully-qualified domain name for historical reasons; other algorithm names are simple (i.e., single-component) names.

IANA maintains a registry of "TSIG Error values" to be used for "Error" values as defined in Section 4.3. Initial values should be those defined in Section 3. New TSIG error codes for the TSIG error field are assigned using the IETF Consensus policy defined in [RFC8126].

11. Security Considerations

The approach specified here is computationally much less expensive than the signatures specified in DNSSEC. As long as the shared secret key is not compromised, strong authentication is provided for the last hop from a local name server to the user resolver.

Secret keys should be changed periodically. If the client host has been compromised, the server should suspend the use of all secrets known to that client. If possible, secrets should be stored in encrypted form. Secrets should never be transmitted in the clear over any network. This document does not address the issue on how to distribute secrets. Secrets should never be shared by more than two entities.

This mechanism does not authenticate source data, only its transmission between two parties who share some secret. The original source data can come from a compromised zone master or can be corrupted during transit from an authentic zone master to some "caching forwarder." However, if the server is faithfully performing the full DNSSEC security checks, then only security checked data will be available to the client.

A fudge value that is too large may leave the server open to replay attacks. A fudge value that is too small may cause failures if machines are not time synchronized or there are unexpected network delays. The recommended value in most situation is 300 seconds.

For all of the message authentication code algorithms listed in this document, those producing longer values are believed to be stronger; however, while there have been some arguments that mild truncation can strengthen a MAC by reducing the information available to an attacker, excessive truncation clearly weakens authentication by

reducing the number of bits an attacker has to try to break the authentication by brute force [RFC2104].

Significant progress has been made recently in cryptanalysis of hash functions of the types used here, all of which ultimately derive from the design of MD4. While the results so far should not effect HMAC, the stronger SHA-1 and SHA-256 algorithms are being made mandatory due to caution. Note that today SHA-3 [FIPS202] is available as an alternative to SHA-2 using a very different design.

See also the Security Considerations section of [RFC2104] from which the limits on truncation in this RFC were taken.

11.1. Issue fixed in this document

When signing a DNS reply message using TSIG, its MAC computation uses the request message's MAC as an input to cryptographically relate the reply to the request. Unfortunately, the original TSIG specification [RFC2845] failed to clearly require the request MAC to be successfully validated before using it.

This document proposes the principle that the MAC must be considered to be invalid until it was validated. This leads to the requirement that only a validated request MAC is included in a signed answer. Or with other words when the request MAC was not validated the answer must be unsigned with a BADKEY or BADSIG TSIG error.

11.2. Why not DNSSEC?

This section from the original document [RFC2845] analyzes DNSSEC in order to justify the introduction of TSIG.

DNS has recently been extended by DNSSEC ([RFC4033], [RFC4034] and [RFC4035]) to provide for data origin authentication, and public key distribution, all based on public key cryptography and public key based digital signatures. To be practical, this form of security generally requires extensive local caching of keys and tracing of authentication through multiple keys and signatures to a pre-trusted locally configured key.

One difficulty with the DNSSEC scheme is that common DNS implementations include simple "stub" resolvers which do not have caches. Such resolvers typically rely on a caching DNS server on another host. It is impractical for these stub resolvers to perform general DNSSEC authentication and they would naturally depend on their caching DNS server to perform such services for them. To do so securely requires secure communication of queries and responses. DNSSEC provides public key transaction signatures to support this,

but such signatures are very expensive computationally to generate. In general, these require the same complex public key logic that is impractical for stubs.

A second area where use of straight DNSSEC public key based mechanisms may be impractical is authenticating dynamic update [RFC2136] requests. DNSSEC provides for request signatures but with DNSSEC they, like transaction signatures, require computationally expensive public key cryptography and complex authentication logic. Secure Domain Name System Dynamic Update ([RFC3007]) describes how different keys are used in dynamically updated zones.

12. References

12.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC4635] Eastlake 3rd, D., "HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers", RFC 4635, DOI 10.17487/RFC4635, August 2006, <https://www.rfc-editor.org/info/rfc4635>.

[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>.

12.2. Informative References

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard", FIPS PUB 202, August 2015.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <https://www.rfc-editor.org/info/rfc2930>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/rfc3007>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, DOI 10.17487/RFC3645, October 2003, <https://www.rfc-editor.org/info/rfc3645>.

[RFC3874] Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, DOI 10.17487/RFC3874, September 2004, <https://www.rfc-editor.org/info/rfc3874>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[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>.

[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>.

Appendix A. Acknowledgments

This document just consolidates and updates the earlier documents by the authors of [RFC2845] (Paul Vixie, Olafur Gudmundsson, Donald E. Eastlake 3rd and Brian Wellington) and [RFC4635] (Donald E. Eastlake 3rd). It would not be possible without their original work.

The security problem addressed by this document was reported by Clement Berthaux from Synacktiv.

Note for the RFC Editor (to be removed before publication): the first ’e’ in Clement is a fact a small ’e’ with acute, unicode code U+00E9. I do not know if xml2rfc supports non ASCII characters so I prefer to

not experiment with it. BTW I am French too too so I can help if you have questions like correct spelling...

Peter van Dijk, Benno Overeinder, Willem Toroop, Ondrej Sury, Mukund Sivaraman and Ralph Dolmans participated in the discussions that prompted this document.

Appendix B. Change History

draft-dupont-dnsop-rfc2845bis-00

[RFC4635] was merged.

Authors of original documents were moved to Acknowledgments (Appendix A).

Section 2 was updated to [RFC8174] style.

Spit references into normative and informative references and updated them.

Added a text explaining why this document was written in the Abstract and at the beginning of the introduction.

Clarified the layout of TSIG RDATA.

Moved the text about using DNSSEC from the Introduction to the end of Security Considerations.

Added the security clarifications:

1. Emphasized that MAC is invalid until it is successfully validated.

2. Added requirement that a request MAC that has not been successfully validated MUST NOT be included into a response.

3. Added requirement that a request that has not been validated to the MUST NOT generate a signed response.

4. Added note about MAC too short for the local policy to the Section 6.3.

5. Changed the order of server checks and swapped corresponding sections.

6. Removed the truncation size limit "also case" as it does not apply and added confusion.

7. Relocated the error provision for TSIG truncation to the new Section 6.5.5. Moved from RCODE 22 to RCODE 9 and TSIG ERROR 22, i.e., aligned with other TSIG error cases.

8. Added Section 6.6.4 about truncation error handling by clients.

9. Removed the limit to HMAC output in replies as a request which specified a MAC length longer than the HMAC output is invalid according the the first processing rule in Section 6.5.2.

10. Promoted the requirement that a secret length should be at least as long as the HMAC output to a SHOULD [RFC2119] key word.

11. Added a short text to explain the security issue.

draft-dupont-dnsop-rfc2845bis-01

Improved wording (post-publication comments).

Specialized and renamed the "TSIG on TCP connection" (Section 6.4) to "TSIG on zone tranfer over a TCP connection". Added a SHOULD for a TSIG in each message (was envelope) for new implementations.

Authors' Addresses

Francis Dupont (editor)
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States

Email: Francis.Dupont@fdupont.fr

Stephen Morris
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States

Email: stephen@isc.org
URI: http://www.isc.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 2, 2018

K. Fujiwara
JPRS

October 29, 2017

Returning additional answers in DNS responses draft-fujiwara-dnsop-additional-answers-00

Abstract

This document proposes to document the ability to provide multiple answers in single DNS response. For example, authoritative servers may add a NSEC resource record or A/AAAA resource records of the query name. This is especially useful as, in many cases, the entity making the request has no a priori knowledge of what other questions it will need to ask. It is already possible (an authoritative server MAY already sends what it wants in the additional section). This document does not propose any protocol changes, just explanations of an already acceptable practice.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 2, 2018.

Table of Contents

1. Introduction

[I-D.wkumari-dnsop-multiple-responses] proposes pseudo resource record that controls resource records added into additional section. It offers any combinations of owner names and record types that are added into additional section.

In many cases, combinations are limited and DNS software developers knows well. This document proposes that DNS server software developers choose the combination of additional data.

By providing multiple answers in single response, authoritative name servers can assist full-service resolvers in pre-populating their cache before stub resolvers or other clients ask for the subsequent queries. Apart from decreasing the latency for end users [RFC6555], this also decreases the total number of queries that full-service resolvers need to send and authoritative servers need to answer.

By providing NSEC/NSEC3 resource record that matches a query name, validating resolvers can generate NODATA or NXDOMAIN responses with Aggressive Use of DNSSEC-validated cache [RFC8198].

Developers of DNS servers know end users' query patterns or full-service resolvers' query patterns well. Authoritative DNS servers may add any authoritative data in the additional section. For example, QTYPE MX queries are followed by mail exchange hosts A/AAAA queries. When an authoritative server receives a QTYPE MX query, some implementations add mail exchange hosts A/AAAA resource records in additional section if the authoritative server have authoritative data of mail exchange hosts.

Other typical examples are A and AAAA, SRV and Target A/AAAA, TLSA RR and corresponding server addresses.

This technique, described in this document, is purely an optimization and enables authoritative servers to distribute some other related answers that the client is likely to need along with an answer to the original request. Users get a better experience, full-service resolvers need to send less queries, authoritative servers have to answer fewer queries, etc.

2. Background

The DNS specifications ([RFC1034], for instance section 4.3.2) allow for supplemental information to be included in the "additional" section of the DNS response, but in order to defeat cache poisoning attacks most implementations either ignore or don't trust additional records they didn't ask for. For more background, see [RFC2181].

Some implementations add mail exchange A/AAAA resource records in MX responses (an actual example is given in section 3.7.1 of [RFC1034]). Some implementations add Target A/AAAA resource records in SRV responses.

3. Terminology

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

Many of the specialized terms used in this specification are defined in DNS Terminology [RFC7719] and [I-D.ietf-dnsop-terminology-bis].

Additional records: Additional records are records that the authoritative nameserver has included in the Additional section.

4. Returning multiple answers

An authoritative nameserver MAY include any additional records that help name resolution. These additional records are appended to the additional section of the response.

To increase the probability that these extra data will actually be useful for the resolver, it is suggested to send them only if:

- The query has DNSSEC OK bit set.
- The authoritative server is authoritative for the additional records, and the records to be returned are DNSSEC signed. The additional records contain RRSIGs.
- To prove the non-existence of the resource record type, additional records may be NSEC/NSEC3 resource records for the query name and some other query names (for example, TLSA owner name). Validating resolvers can generate negative NODATA/NXDOMAIN response with Aggressive Use of DNSSEC-validated cache [RFC8198].
- Responses with additional records fit in the required response size.

5. Possible additional answers

Possible query and additional records pairs are:

- NAME A : NAME AAAA (or NAME NSEC/NSEC3)
- NAME AAAA : NAME A (or NAME NSEC/NSEC3)
- NAME MX : mail exchange A/AAAA (and/or mail exchange NSEC/NSEC3)
- NAME SRV : Target host A/AAAA (and/or Target host NSEC/NSEC3)
- NAME A/AAAA : _443._tcp.NAME TLSA (and/or NAME NSEC/NSEC3)
- _443._tcp.NAME TLSA : NAME A/AAAA (and/or NAME NSEC/NSEC3)

TLSA / MX / SRV pairs have different query names.

6. Stub-Resolver Considerations

No modifications need to be made to stub-resolvers to get the predominate benefit of this protocol, since the majority of the speed gain will take place between the validating recursive resolver and the authoritative name server. However, stub resolvers and full-

service resolvers may use this technique if stub-resolvers are validating stub resolvers.

7. Use of Additional information

When deciding to use additional records in the additional section, a resolver should follow certain rules:

- Additional records are validated before being used.
- Additional records SHOULD have lower priority in the cache than answers received because they were requested. This is to help evict Additional records from the cache first (to help prevent cache filling attacks).
- Recursive resolvers MAY choose to ignore Additional records for any reason, including CPU or cache space concerns, phase of the moon, etc. It may choose to accept all, some or none of the Additional record sets.
- Recursive resolvers SHOULD support "Aggressive use of DNSSEC-validated cache" [RFC8198].

These rules are derived from [RFC2181] and DNSSEC RFCs.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

The use of DNSSEC guarantees that these additional records will be accepted and cached by the resolver only if they can be proved genuine.

The technique described in this document makes DNS response size large. If DNS response size exceeds path MTU, the response will be fragmented and the fragmentation may cause problems. Authoritative DNS server software developers and operators need to choose suitable response size limit.

10. Acknowledgments

The author acknowledges authors of [I-D.wkumari-dnsop-multiple-responses] because many part of idea and texts are copied from the draft.

The author would like to specifically thank Stephane Bortzmeyer for extensive review and comments.

11. Change History

12. References

12.1. Normative References

[I-D.ietf-dnsop-terminology-bis]
Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", draft-ietf-dnsop-terminology-bis-07 (work in progress), October 2017.

[I-D.wkumari-dnsop-multiple-responses]
Kumari, W., Yan, Z., Hardaker, W., and D. Lawrence, "Returning extra answers in DNS responses.", draft-wkumari-dnsop-multiple-responses-05 (work in progress), July 2017.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

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

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <https://www.rfc-editor.org/info/rfc6555>.

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

[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>.

12.2. Informative References

[I-D.bellis-dnsext-multi-qtypes]
Bellis, R., "DNS Multiple QTYPEs", draft-bellis-dnsext-multi-qtypes-04 (work in progress), July 2017.

[I-D.yao-dnsop-accompanying-questions]
Yao, J., Vixie, P., Kong, N., and X. Lee, "A DNS Query including A Main Question with Accompanying Questions", draft-yao-dnsop-accompanying-questions-04 (work in progress), September 2017.

Appendix A. Comparisons of multiple response proposals

A.1. draft-wkumari-dnsop-multiple-responses

[I-D.wkumari-dnsop-multiple-responses] proposes pseudo resource record that controls resource records added into additional section.

No protocol changes between authoritative servers and full-service resolvers. New authoritative server software required. Zone operators need to configure. Supports different owner names and types. Answer size becomes large if the query matches operators configuration. Requires DNSSEC.

A.2. draft-fujiwara-dnsop-additional-answers

draft-fujiwara-dnsop-additional-answers proposes that authoritative servers add well used additional records and NSEC/NSEC3 resource records in additional section.

No protocol changes between authoritative servers and full-service resolvers. New authoritative server software required. No configuration. Supports different owner names and types. Answer size becomes large (always). Requires DNSSEC and [RFC8198].

A.3. draft-bellis-dnsext-multi-qtypes

[I-D.bellis-dnsext-multi-qtypes] proposes new EDNS options that carry additional query types.

New authoritative server software required. New full-service resolver software required. No configuration. No support of different owner names.

A.4. draft-yao-dnsop-accompanying-questions

[I-D.yao-dnsop-accompanying-questions] proposes new EDNS option that carry additional query names, query types and rcodes.

New authoritative server software required. New full-service resolver software required. No configuration.

A.5. QDCOUNT>1 idea

No drafts. QDCOUNT is not limited to 1 in [RFC1035].

No protocol changes between authoritative servers and full-service resolvers, however, some implementations (For example, BIND 9, NSD, Unbound) treats QDCOUNT>1 as FORMERR. New authoritative server software required. New full-service resolver software required. Supports different owner names and types, however, it cannot answer different rcodes. No configuration. A database that each IP address support QDCOUNT>1 is required in full-service resolvers.

A.6. Comparison chart

Draft	wkumari	fujiawra	bellis	yao	QDCOUNT>1
Protocol change	No	No	Yes	Yes	Yes
New Auth soft code size	Yes some	Yes little	Yes large?	Yes large?	Yes large?
New Resolver soft	No	No	Required	Required	Required
Config complexity	Yes	No	No	No	No
Multiple names	Yes	Yes	No	Yes	maybe
Multiple types	Yes	Yes	Yes	Yes	Yes
Multiple rcodes	---	---	need not	Yes	No
Require DNSSEC	(Yes)	(Yes)	No	No	No
Response fat if	config	always	query	query	query
Stub support ?	No	No	possible	possible	possible
IP addr Database	No	No	EDNS	EDNS	New
Deploy?	Easy	Easy	Yes?	Yes?	No?

Author’s Address

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 14, 2018

K. Fujiwara
JPRS

January 10, 2018

Returning additional answers in DNS responses draft-fujiwara-dnsop-additional-answers-01

Abstract

This document proposes to document the ability to provide multiple answers in single DNS response. For example, authoritative servers may add a NSEC resource record or A/AAAA resource records of the query name. This is especially useful as, in many cases, the entity making the request has no a priori knowledge of what other questions it will need to ask. It is already possible (an authoritative server MAY already sends what it wants in the additional section). This document does not propose any protocol changes, just explanations of an already acceptable practice.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 14, 2018.

Table of Contents

1. Introduction

[I-D.wkumari-dnsop-multiple-responses] proposes pseudo resource record that controls resource records added into additional section. It offers any combinations of owner names and record types that are added into additional section.

In many cases, combinations are limited and DNS software developers knows well. This document proposes that DNS server software developers choose the combination of additional data.

By providing multiple answers in single response, authoritative name servers can assist full-service resolvers in pre-populating their cache before stub resolvers or other clients ask for the subsequent queries. Apart from decreasing the latency for end users [RFC6555],

this also decreases the total number of queries that full-service resolvers need to send and authoritative servers need to answer.

By providing NSEC/NSEC3 resource record that matches a query name, validating resolvers can generate NODATA or NXDOMAIN responses with Aggressive Use of DNSSEC-validated cache [RFC8198].

Developers of DNS servers know end users' query patterns or full-service resolvers' query patterns well. Authoritative DNS servers may add any authoritative data in the additional section. For example, QTYPE MX queries are followed by mail exchange hosts A/AAAA queries. When an authoritative server receives a QTYPE MX query, some implementations add mail exchange hosts A/AAAA resource records in additional section if the authoritative server have authoritative data of mail exchange hosts.

Other typical examples are A and AAAA, SRV and Target A/AAAA, TLSA RR and corresponding server addresses.

This technique, described in this document, is purely an optimization and enables authoritative servers to distribute some other related answers that the client is likely to need along with an answer to the original request. Users get a better experience, full-service resolvers need to send less queries, authoritative servers have to answer fewer queries, etc.

2. Background

The DNS specifications ([RFC1034], for instance section 4.3.2) allow for supplemental information to be included in the "additional" section of the DNS response, but in order to defeat cache poisoning attacks most implementations either ignore or don't trust additional records they didn't ask for. For more background, see [RFC2181].

Some implementations add mail exchange A/AAAA resource records in MX responses (an actual example is given in section 3.7.1 of [RFC1034]). Some implementations add Target A/AAAA resource records in SRV responses.

3. Terminology

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

Many of the specialized terms used in this specification are defined in DNS Terminology [RFC7719] and [I-D.ietf-dnsop-terminology-bis].

Additional records: Additional records are records that the authoritative nameserver has included in the Additional section.

4. Returning multiple answers

An authoritative nameserver MAY include any additional records that help name resolution. These additional records are appended to the additional section of the response.

To increase the probability that these extra data will actually be useful for the resolver, it is suggested to send them only if:

- The query has DNSSEC OK bit set.
- The authoritative server is authoritative for the additional records, and the records to be returned are DNSSEC signed. The additional records contain RRSIGs.
- To prove the non-existence of the resource record type, additional records may be NSEC/NSEC3 resource records for the query name and some other query names (for example, TLSA owner name). Validating resolvers can generate negative NODATA/NXDOMAIN response with Aggressive Use of DNSSEC-validated cache [RFC8198].
- Responses with additional records fit in the required response size.

Additional records may be controlled by server configuration. "enable additional a/aaaa" or "enable additonal nsec*" options are possible.

5. Possible additional answers

Possible query and additional records pairs are:

- NAME A : NAME AAAA (or NAME NSEC/NSEC3)
- NAME AAAA : NAME A (or NAME NSEC/NSEC3)
- NAME MX : mail exchange A/AAAA (and/or mail exchange NSEC/NSEC3)
- NAME SRV : Target host A/AAAA (and/or Target host NSEC/NSEC3)
- NAME A/AAAA : _443._tcp.NAME TLSA (and/or NAME NSEC/NSEC3)
- _443._tcp.NAME TLSA : NAME A/AAAA (and/or NAME NSEC/NSEC3)

TLSA / MX / SRV pairs have different query names.

6. Stub-Resolver Considerations

No modifications need to be made to stub-resolvers to get the predominate benefit of this protocol, since the majority of the speed gain will take place between the validating recursive resolver and the authoritative name server. However, stub resolvers and full-service resolvers may use this technique if stub-resolvers are validating stub resolvers.

7. Use of Additional information

When deciding to use additional records in the additional section, a resolver should follow certain rules:

- Additional records are validated before being used.
- Additional records SHOULD have lower priority in the cache than answers received because they were requested. This is to help evict Additional records from the cache first (to help prevent cache filling attacks).
- Recursive resolvers MAY choose to ignore Additional records for any reason, including CPU or cache space concerns, phase of the moon, etc. It may choose to accept all, some or none of the Additional record sets.
- Recursive resolvers SHOULD support "Aggressive use of DNSSEC-validated cache" [RFC8198].

These rules are derived from [RFC2181] and DNSSEC RFCs.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

The use of DNSSEC guarantees that these additional records will be accepted and cached by the resolver only if they can be proved genuine.

The technique described in this document makes DNS response size large. If DNS response size exceeds path MTU, the response will be fragmented and the fragmentation may cause problems. Authoritative DNS server software developers and operators need to choose suitable response size limit.

10. Acknowledgments

The author acknowledges authors of [I-D.wkumari-dnsop-multiple-responses] because many part of idea and texts are copied from the draft.

The author would like to specifically thank Stephane Bortzmeyer for extensive review and comments.

11. Change History

11.1. 00 to 01

Sync with IETF 100 presentation

- Added system wide configration that controls additional records
- Added "draft-vavrusa-dnsop-aaaa-for-free"
- Updated comparison table

12. References

12.1. Normative References

[I-D.ietf-dnsop-terminology-bis] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", draft-ietf-dnsop-terminology-bis-08 (work in progress), November 2017.

[I-D.wkumari-dnsop-multiple-responses] Kumari, W., Yan, Z., Hardaker, W., and D. Lawrence, "Returning extra answers in DNS responses.", draft-wkumari-dnsop-multiple-responses-05 (work in progress), July 2017.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

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

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <https://www.rfc-editor.org/info/rfc6555>.

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

[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>.

12.2. Informative References

[I-D.bellis-dnsext-multi-qtypes]
Bellis, R., "DNS Multiple QTYPEs", draft-bellis-dnsext-multi-qtypes-05 (work in progress), January 2018.

[I-D.vavrusa-dnsop-aaaa-for-free]
marek@vavrusa.com, m. and O. Gu[eth]mundsson, "Providing AAAA records for free with QTYPE=A", draft-vavrusa-dnsop-aaaa-for-free-00 (work in progress), March 2016.

[I-D.yao-dnsop-accompanying-questions]
Yao, J., Vixie, P., Kong, N., and X. Lee, "A DNS Query including A Main Question with Accompanying Questions", draft-yao-dnsop-accompanying-questions-04 (work in progress), September 2017.

Appendix A. Comparisons of multiple response proposals

A.1. draft-wkumari-dnsop-multiple-responses

[I-D.wkumari-dnsop-multiple-responses] proposes pseudo resource record that controls resource records added into additional section.

No protocol changes between authoritative servers and full-service resolvers. New authoritative server software required. Zone operators need to configure. Supports different owner names and types. Answer size becomes large if the query matches operators configuration. Requires DNSSEC.

A.2. draft-fujiwara-dnsop-additional-answers

draft-fujiwara-dnsop-additional-answers proposes that authoritative servers add well used additional records and NSEC/NSEC3 resource records in additional section.

No protocol changes between authoritative servers and full-service resolvers. New authoritative server software required. No configuration. Supports different owner names and types. Answer size becomes large (always). Requires DNSSEC and [RFC8198].

A.3. draft-bellis-dnsext-multi-qtypes

[I-D.bellis-dnsext-multi-qtypes] proposes new EDNS options that carry additional query types.

New authoritative server software required. New full-service resolver software required. No configuration. No support of different owner names.

A.4. draft-yao-dnsop-accompanying-questions

[I-D.yao-dnsop-accompanying-questions] proposes new EDNS option that carry additional query names, query types and rcodes.

New authoritative server software required. New full-service resolver software required. No configuration.

A.5. draft-vavrusa-dnsop-aaaa-for-free

[I-D.vavrusa-dnsop-aaaa-for-free] proposes additional AAAA resource records in answer section. New authoritative server software required. New full-service resolver software required because existing full-service resolvers ignore additional AAAA resource records. No configuration.

A.6. QDCOUNT>1 idea

No drafts. QDCOUNT is not limited to 1 in [RFC1035].

No protocol changes between authoritative servers and full-service resolvers, however, some implementations (For example, BIND 9, NSD, Unbound) treats QDCOUNT>1 as FORMERR. New authoritative server software required. New full-service resolver software required. Supports different owner names and types, however, it cannot answer different rcodes. No configuration. A database that each IP address support QDCOUNT>1 is required in full-service resolvers.

A.7. Comparison chart

Draft	additional answers	multiple responses	aaaa for free	multi qtypes	accompanying querstions
Protocol change	No	No	Yes?	Yes	Yes
Code size	little	some	little	large?	large?
Resolver modification	No	No	Yes?	Yes	Yes
Config complexity	No	Yes	No	No	No
Multiple names	Yes	Yes	No	No	Yes
Multiple types	Yes	Yes	AAAA	Yes	Yes
Multiple rcodes	(NSEC*)	---	---	---	Yes
Negative response	Yes	No	No	Yes	Yes
Fat response if	always	config	always	query	query
Stub support ?	No	No	?	possible	possible
Deployment	easy	easy	gradual	gradual	gradual
Require DNSSEC	(Yes)	(Yes)	No	No	No
IP addr Database	No	No	No	EDNS	EDNS

Author’s Address

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

DNSOP
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2018

G. Huston
J. Damas
APNIC
W. Kumari
Google

October 26, 2017

A Sentinel for Detecting Trusted Keys in DNSSEC
draft-huston-kskroll-sentinel-02.txt

Abstract

The DNS Security Extensions (DNSSEC) were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. These digital signatures can be verified by building a chain of trust starting from a trust anchor and proceeding down to a particular node in the DNS. This document specifies a mechanism that will allow an end user to determine the trusted key state of the resolvers that handle the user's DNS queries.

Table of Contents

1. Introduction

The DNS Security Extensions (DNSSEC) [RFC4033], [RFC4034] and [RFC4035] were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. DNSSEC uses Key Tags to efficiently match signatures to the keys from which they are generated. The Key Tag is a 16-bit value computed from the RDATA portion of a DNSKEY RR using a formula not unlike a ones-complement checksum. RRSIG RRs contain a Key Tag field whose value is equal to the Key Tag of the DNSKEY RR that validates the signature.

This document specifies how validating resolvers can respond to certain queries in a manner that allows a querier to deduce whether a particular key has been loaded into that resolver's trusted key store. In particular, this response mechanism can be used to determine whether a certain Root Zone KSK is ready to be used as a trusted key within the context of a key roll by this resolver.

This new mechanism is OPTIONAL to implement and use, although for reasons of supporting broad-based measurement techniques, it is strongly preferred if configurations of DNSSEC-validating resolvers enabled this mechanism by default, allowing for configuration directives to disable this mechanism if desired.

1.1. Terminology

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

2. Sentinel Mechanism

DNSSEC-Validating resolvers that implement this mechanism MUST be performing validation of responses in accordance with the DNSSEC response validation specification [RFC4035].

This mechanism makes use of 2 special labels, "._is-ta-<tag-index>." (Intended to be used in a query where the response can answer the question: Is this the key tag a trust anchor which the validating DNS resolver is currently trusting?) and "._not-ta-<tag-index>." (Intended to be used in a query where the response can answer the question: Is this the key tag of a key that is NOT in the resolver's current trust store?). The use of the positive question and its inverse allows for queries to detect whether resolvers support this mechanism.

If the outcome of the DNS response validation process indicates that the response is authentic, and if the original query contains exactly one label that matches the template "._is-ta-<tag-index>.", then the following rule should be applied to the response: If the resolver has placed a Root Zone Key Signing Key with tag index value matching the value specified in the query into the local resolver's store of trusted keys, then the resolver should return a response indicating that the response contains authenticated data according to section 5.8 of [RFC6840]. Otherwise, the resolver MUST return RCODE 2 (server failure). Note that the <tag-index> is specified in the DNS label using hex notation.

If the outcome of the DNS response validation process indicates that the response is authentic, and if the original query contains exactly one label that matches the template "._not-ta-<tag-index>.", then the following rule should be applied to the response: If the resolver has not placed a Root Zone Key Signing Key with tag index value matching the value specified in the query into the local resolver's store of trusted keys, then the resolver should return a response indicating that the response contains authenticated data according to section 5.8 of [RFC6840]. Otherwise, the resolver MUST return RCODE 2 (server failure). Note that the <tag-index> is specified in the DNS label using hex notation.

If a query contains one instance of both of these query templates then the resolver MUST NOT alter the outcome of the DNS response validation process.

This mechanism is to be applied only by resolvers that perform DNSSEC validation, and applies only to responses to an A or AAAA query (Query Type value 1 or 28) where the resolver has authenticated the response according to the DNSSEC validation process and where the query name contains either of the labels described in this section. In this case, the resolver is to perform an additional test following the conventional validation function as described in this section. The result of this test directs whether the resolver is to change an authentic response to a response that indicates validation failure.

3. Sentinel Processing

This proposed test that uses the DNS resolver mechanism described in this document is based on three DNS names that have three distinct DNS resolution behaviours. The test is intended to allow a user to determine the state of their DNS resolution system, and, in particular, whether or not they are using validating DNS resolvers that have picked up an incoming trust anchor in a key roll.

The name format can be defined in a number of ways, and no name form is intrinsically better than any other in terms of the test itself. The critical aspect of the DNS names used in any such test is that they contain the specified label for either the positive and negative test.

The sentinel process is envisaged to use a test with three names:

a. a name containing the label "._is-ta-<tag-index>.". This is a validly signed name so that responses about names in this zone can be authenticated by a validating resolver.

b. a name containing the label "._not-ta-<tag-index>.". This is also a validly-signed name.

c. a third name that is signed with a DNSSEC signature that cannot be validated.

The responses received from queries to resolve each of these names would allow us to infer a trust key state of the resolution environment.

o Vnew: A DNSSEC-Validating resolver that includes this mechanism that has loaded the nominated key into its trusted key stash will

respond with an A record response for "is-ta", SERVFAIL for "not-ta" and SERVFAIL for the invalid name.

- Vold: A DNSSEC-Validating resolver that includes this mechanism that has not loaded the nominated key into its trusted key stash will respond with an SERVFAIL record for "is-ta", an A record response for "not-ta" and SERVFAIL for the invalid name.

- Vleg: A DNSSEC-Validating resolver that does not include this mechanism will respond with an A record response for "is-ta", an A record response for "not-ta" and SERVFAIL for the invalid name.

- nonV: A non-DNSSEC-Validating resolver will respond with an A record response for "is-ta", an A record response for "not-ta" and an A record response for the invalid name.

Given the clear delineation amongst these three cases, if a client directs these three queries to a simple resolver, the variation in response to the three queries should allow the client to determine the category of the resolver, and if it supports this mechanism, whether or not it has loaded a particular key into its local trusted key stash.

Type\Query	is_ta	not_ta	invalid
Vnew	A	SERVFAIL	SERVFAIL
Vold	SERVFAIL	A	SERVFAIL
Vleg	A	A	SERVFAIL
nonV	A	A	A

A Vnew response pattern says that the nominated key is trusted by the resolver and has been loaded into its local trusted key stash. A Vleg response pattern says that the nominated key is not yet trusted by the resolver in its own right. A Vleg response is indeterminate, and a nonV response indicates that the client does not have a validating resolver.

4. Sentinel Test Result Considerations

The description in the previous section describes a simple situation where the test queries were being passed to a single recursive resolver that directly queried authoritative name servers, including the root servers.

There is also the common case where the end client is configured to use multiple resolvers. In these cases the SERVFAIL responses from one resolver will prompt the end client to repeat the query against one of the other configured resolvers.

If any of the client's resolvers are non-validating resolvers, the tests will result in the client reporting that it has a non-validating DNS environment (nonV), which is effectively the case.

If all of the client resolvers are DNSSEC-validating resolvers, but some do not support this trusted key mechanism, then the result will be indeterminate with respect to trusted key status (Vleg). Simlarly, if all the client's resolvers support this mechanism, but some have loaded the key into the trusted key stash and some have not, then the result is indeterminate (Vleg).

There is also the common case of a recursive resolver using a forwarder.

If the resolver is non-validating, and it has a single forwarder clause, then the resolver will presumably mirror the capabilities of the forwarder target resolver. If this non-validating resolver it has multiple forwarders, then the above considerations will apply.

If the validating resolver has a forwarding configuration, and uses the CD flag on all forwarded queries, then this resolver is acting in a manner that is identical to a standalone resolver. The same consideration applies if any one one of the forwarder targets is a non-validating resolver. Similarly, if all the forwarder targets do not apply this trusted key mechanism, the same considerations apply.

A more complex case is where the following conditions all hold:

- both the validating resolver and the forwarder target resolver support this trusted key sentinel mechanism, and
- the local resolver's queries do not carry the CD bit, and
- the trusted key state differs between the forwarding resolver and the forwarder target resolver

then either the outcome is indeterminate validating (Vleg), or a case of mixed signals (SERVFAIL in all three responses), which is similarly an indeterminate response with respect to the trusted key state.

5. Security Considerations

This document describes a mechanism to allow users to determine the trust state of root zone key signing keys in the DNS resolution system that they use.

The mechanism does not require resolvers to set otherwise unauthenticated responses to be marked as authenticated, and does not alter the security properties of DNSSEC with respect to the interpretation of the authenticity of responses that are so marked.

The mechanism does not require any further significant processing of DNS responses, and queries of the form described in this document do not impose any additional load that could be exploited in an attack over the the normal DNSSEC validation processing load.

6. IANA Considerations

[Note to IANA, to be removed prior to publication: there are no IANA considerations stated in this version of the document.]

7. Acknowledgements

This document has borrowed extensively from RFC8145 for the introductory text, and the authors would like to acknowledge and thank the authors of that document both for some text excerpts and for the more general stimulation of thoughts about monitoring the progress of a roll of the Key Signing Key of the Root Zone of the DNS.

8. References

8.1. Normative References

[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>.

[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>.

8.2. Informative References

[RFC8145] Wessels, D., Kumari, W., and P. Hoffman, "Signaling Trust Anchor Knowledge in DNS Security Extensions (DNSSEC)", RFC 8145, DOI 10.17487/RFC8145, April 2017, <https://www.rfc-editor.org/info/rfc8145>.

Authors' Addresses

Geoff Huston

Email: gih@apnic.net
URI: http://www.apnic.net

Joao Silva Damas

Email: joao@apnic.net
URI: http://www.apnic.net

Warren Kumari

Email: warren@kumari.net

DNSOP
Internet-Draft
Intended status: Standards Track
Expires: May 18, 2018

G. Huston
J. Damas
APNIC
W. Kumari
Google

November 14, 2017

A Sentinel for Detecting Trusted Keys in DNSSEC
draft-huston-kskroll-sentinel-04.txt

Abstract

The DNS Security Extensions (DNSSEC) were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. These digital signatures can be verified by building a chain of trust starting from a trust anchor and proceeding down to a particular node in the DNS. This document specifies a mechanism that will allow an end user to determine the trusted key state of the resolvers that handle the user's DNS queries.

Table of Contents

1. Introduction

The DNS Security Extensions (DNSSEC) [RFC4033], [RFC4034] and [RFC4035] were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. DNSSEC uses Key Tags to efficiently match signatures to the keys from which they are generated. The Key Tag is a 16-bit value computed from the RDATA portion of a DNSKEY RR using a formula not unlike a ones-complement checksum. RRSIG RRs contain a Key Tag field whose value is equal to the Key Tag of the DNSKEY RR that validates the signature.

This document specifies how validating resolvers can respond to certain queries in a manner that allows a querier to deduce whether a particular key has been loaded into that resolver's trusted key store. In particular, this response mechanism can be used to determine whether a certain Root Zone KSK is ready to be used as a trusted key within the context of a key roll by this resolver.

This new mechanism is OPTIONAL to implement and use, although for reasons of supporting broad-based measurement techniques, it is strongly preferred if configurations of DNSSEC-validating resolvers enabled this mechanism by default, allowing for configuration directives to disable this mechanism if desired.

1.1. Terminology

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

2. Sentinel Mechanism

DNSSEC-Validating resolvers that implement this mechanism MUST be performing validation of responses in accordance with the DNSSEC response validation specification [RFC4035].

This mechanism makes use of 2 special labels, "_is-ta-<tag-index>." (Intended to be used in a query where the response can answer the question: Is this the key tag a trust anchor which the validating DNS resolver is currently trusting?) and "_not-ta-<tag-index>." (Intended to be used in a query where the response can answer the question: Is this the key tag of a key that is NOT in the resolver's current trust store?). The use of the positive question and its inverse allows for queries to detect whether resolvers support this mechanism.

If the outcome of the DNS response validation process indicates that the response is authentic, and if the left-most label of the original query name matches the template "_is-ta-<tag-index>.", then the following rule should be applied to the response: If the resolver has placed a Root Zone Key Signing Key with tag index value matching the value specified in the query into the local resolver's store of trusted keys, then the resolver should return a response indicating that the response contains authenticated data according to section 5.8 of [RFC6840]. Otherwise, the resolver MUST return RCODE 2 (server failure). Note that the <tag-index> is specified in the DNS label using hex notation.

If the outcome of the DNS response validation process indicates that the response is authentic, and if the left-most label of the qriginal query name matches the template "_not-ta-<tag-index>.", then the following rule should be applied to the response: If the resolver has not placed a Root Zone Key Signing Key with tag index value matching the value specified in the query into the local resolver's store of trusted keys, then the resolver should return a response indicating that the response contains authenticated data according to section 5.8 of [RFC6840]. Otherwise, the resolver MUST return RCODE 2 (server failure). Note that the <tag-index> is specified in the DNS label using hex notation.

If a query contains one instance of both of these query templates then the resolver MUST NOT alter the outcome of the DNS response validation process.

This mechanism is to be applied only by resolvers that perform DNSSEC validation, and applies only to responses to an A or AAAA query (Query Type value 1 or 28) where the resolver has authenticated the response according to the DNSSEC validation process and where the query name contains either of the labels described in this section. In this case, the resolver is to perform an additional test following the conventional validation function as described in this section. The result of this test directs whether the resolver is to change an authentic response to a response that indicates validation failure.

3. Sentinel Processing

This proposed test that uses the DNS resolver mechanism described in this document is based on three DNS names that have three distinct DNS resolution behaviours. The test is intended to allow a user to determine the state of their DNS resolution system, and, in particular, whether or not they are using validating DNS resolvers that have picked up an incoming trust anchor in a key roll.

The name format can be defined in a number of ways, and no name form is intrinsically better than any other in terms of the test itself. The critical aspect of the DNS names used in any such test is that they contain the specified label for either the positive and negative test.

The sentinel process is envisaged to use a test with three names:

a. a name containing the left-most label "_is-ta-<tag-index>.". This is a validly signed name so that responses about names in this zone can be authenticated by a validating resolver.

b. a name containing the left-most label "_not-ta-<tag-index>.". This is also a validly-signed name.

c. a third name that is signed with a DNSSEC signature that cannot be validated.

The responses received from queries to resolve each of these names would allow us to infer a trust key state of the resolution environment.

o Vnew: A DNSSEC-Validating resolver that includes this mechanism that has loaded the nominated key into its trusted key stash will

respond with an A record response for "_is-ta", SERVFAIL for "_not-ta" and SERVFAIL for the invalid name.

- Vold: A DNSSEC-Validating resolver that includes this mechanism that has not loaded the nominated key into its trusted key stash will respond with an SERVFAIL record for "_is-ta", an A record response for "_not-ta" and SERVFAIL for the invalid name.

- Vleg: A DNSSEC-Validating resolver that does not include this mechanism will respond with an A record response for "_is-ta", an A record response for "_not-ta" and SERVFAIL for the invalid name.

- nonV: A non-DNSSEC-Validating resolver will respond with an A record response for "_is-ta", an A record response for "_not-ta" and an A record response for the invalid name.

Given the clear delineation amongst these three cases, if a client directs these three queries to a simple resolver, the variation in response to the three queries should allow the client to determine the category of the resolver, and if it supports this mechanism, whether or not it has loaded a particular key into its local trusted key stash.

Type\Query	_is-ta	_not-ta	invalid
Vnew	A	SERVFAIL	SERVFAIL
Vold	SERVFAIL	A	SERVFAIL
Vleg	A	A	SERVFAIL
nonV	A	A	A

A Vnew response pattern says that the nominated key is trusted by the resolver and has been loaded into its local trusted key stash. A Vleg response pattern says that the nominated key is not yet trusted by the resolver in its own right. A Vleg response is indeterminate, and a nonV response indicates that the client does not have a validating resolver.

4. Sentinel Test Result Considerations

The description in the previous section describes a simple situation where the test queries were being passed to a single recursive resolver that directly queried authoritative name servers, including the root servers.

There is also the common case where the end client is configured to use multiple resolvers. In these cases the SERVFAIL responses from one resolver will prompt the end client to repeat the query against one of the other configured resolvers.

If any of the client's resolvers are non-validating resolvers, the tests will result in the client reporting that it has a non-validating DNS environment (nonV), which is effectively the case.

If all of the client resolvers are DNSSEC-validating resolvers, but some do not support this trusted key mechanism, then the result will be indeterminate with respect to trusted key status (Vleg). Simlarly, if all the client's resolvers support this mechanism, but some have loaded the key into the trusted key stash and some have not, then the result is indeterminate (Vleg).

There is also the common case of a recursive resolver using a forwarder.

If the resolver is non-validating, and it has a single forwarder clause, then the resolver will presumably mirror the capabilities of the forwarder target resolver. If this non-validating resolver it has multiple forwarders, then the above considerations will apply.

If the validating resolver has a forwarding configuration, and uses the CD flag on all forwarded queries, then this resolver is acting in a manner that is identical to a standalone resolver. The same consideration applies if any one one of the forwarder targets is a non-validating resolver. Similarly, if all the forwarder targets do not apply this trusted key mechanism, the same considerations apply.

A more complex case is where the following conditions all hold:

- both the validating resolver and the forwarder target resolver support this trusted key sentinel mechanism, and
- the local resolver's queries do not carry the CD bit, and
- the trusted key state differs between the forwarding resolver and the forwarder target resolver

then either the outcome is indeterminate validating (Vleg), or a case of mixed signals (SERVFAIL in all three responses), which is similarly an indeterminate response with respect to the trusted key state.

5. Security Considerations

This document describes a mechanism to allow users to determine the trust state of root zone key signing keys in the DNS resolution system that they use.

The mechanism does not require resolvers to set otherwise unauthenticated responses to be marked as authenticated, and does not alter the security properties of DNSSEC with respect to the interpretation of the authenticity of responses that are so marked.

The mechanism does not require any further significant processing of DNS responses, and queries of the form described in this document do not impose any additional load that could be exploited in an attack over the the normal DNSSEC validation processing load.

6. IANA Considerations

[Note to IANA, to be removed prior to publication: there are no IANA considerations stated in this version of the document.]

7. Acknowledgements

This document has borrowed extensively from RFC8145 for the introductory text, and the authors would like to acknowledge and thank the authors of that document both for some text excerpts and for the more general stimulation of thoughts about monitoring the progress of a roll of the Key Signing Key of the Root Zone of the DNS.

8. References

8.1. Normative References

[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>.

[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>.

8.2. Informative References

[RFC8145] Wessels, D., Kumari, W., and P. Hoffman, "Signaling Trust Anchor Knowledge in DNS Security Extensions (DNSSEC)", RFC 8145, DOI 10.17487/RFC8145, April 2017, <https://www.rfc-editor.org/info/rfc8145>.

Authors' Addresses

Geoff Huston

Email: gih@apnic.net
URI: http://www.apnic.net

Joao Silva Damas

Email: joao@apnic.net
URI: http://www.apnic.net

Warren Kumari

Email: warren@kumari.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2018

W. Kumari
Google
E. Hunt
ISC
R. Arends
Nominet
W. Hardaker
USC/ISI
D. Lawrence
Akamai Technologies
October 16, 2017

Extended DNS Errors
draft-ietf-dnsop-extended-error-00

Abstract

This document defines an extensible method to return additional information about the cause of DNS errors. The primary use case is to extend SERVFAIL to provide additional information about the cause of DNS and DNSSEC failures.

[Open question: The document currently defines a registry for errors. It has also been suggested that the option also carry human readable (text) messages, to allow the server admin to provide additional debugging information (e.g: "example.com pointed their NS at us. No idea why...", "We don't provide recursive DNS to 192.0.2.0. Please stop asking...", "Have you tried Acme Anvil and DNS? We do DNS right..." (!). Please let us know if you think text is needed, or if a 16bit FCFS registry is expressive enough.]

[Open question: This document discusses extended *errors*, but it has been suggested that this could be used to also annotate *non-error* messages. The authors do not think that this is a good idea, but could be persuaded otherwise.]

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction and background

There are many reasons that a DNS query may fail, some of them transient, some permanent; some can be resolved by querying another server, some are likely best handled by stopping resolution. Unfortunately, the error signals that a DNS server can return are very limited, and are not very expressive. This means that applications and resolvers often have to "guess" at what the issue is - e.g the answer was marked REFUSED because of a lame delegation, or because of a lame delegation or because the nameserver is still starting up and loading zones? Is a SERVFAIL a DNSSEC validation issue, or is the nameserver experiencing a bad hair day?

A good example of issues that would benefit by additional error information is an error caused by a DNSSEC validation issue. When a stub resolver queries a DNSSEC bogus name (using a validating resolver), the stub resolver receives only a SERVFAIL in response. Unfortunately, SERVFAIL is used to signal many sorts of DNS errors, and so the stub resolver simply asks the next configured DNS resolver. The result of trying the next resolver is one of two outcomes: either the next resolver also validates, a SERVFAIL is returned again, and the user gets an (largely) incomprehensible error message; or the next resolver is not a validating resolver, and the user is returned a potentially harmful result.

This document specifies a mechanism to extend (or annotate) DNS errors to provide additional information about the cause of the error. This information can be used by the resolver to make a decision regarding whether or not to retry, or by technical users attempting to debug issues.

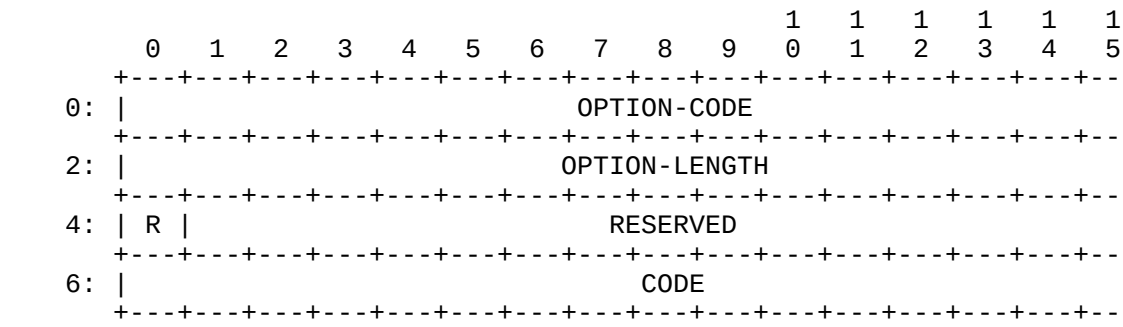
Here is a reference to an "external" (non-RFC / draft) thing: ([IANA.AS_Numbers]). And this is a link to an ID:[I-D.ietf-sidr-iana-objects].

1.1. Requirements notation

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

2. Extended Error EDNS0 option format

This draft uses an EDNS0 ([RFC2671]) option to include extended error (ExtError) information in DNS messages. The option is structured as follows:



- OPTION-CODE, 2 octets (defined in [RFC6891]), for ExtError is TBD.

- OPTION-LENGTH, 2 octets ((defined in [RFC6891]) contains the length of the payload (everything after OPTION-LENGTH) in octets and should be 4.

- RESERVED, 2 octets; the first bit (R) indicates a flag defined in this specification. The remaining bits are reserved for future use, potentially as additional flags.

- CODE, 2 octets.

Currently the only defined flag is the R flag.

R - Retry The R (or Retry) flag provides a hint to the receiver that it should retry the query, probably by querying another server. If the R bit is set (1), the sender believes that retrying the query may provide a successful answer next time; if the R bit is clear (0), the sender believes that it should not ask another server.

The remaining bits in the RESERVED field are reserved for future use and MUST be set to 0 by the sender and SHOULD be ignored by the receiver.

Code: A code point into the IANA "Extended DNS Errors" registry.

3. Use of the Extended DNS Error option

The Extended DNS Error (EDE) is an EDNS option. It can be included in any error response (SERVFAIL, NXDOMAIN, REFUSED, etc) to a query that includes an EDNS option. This document includes a set of initial codepoints (and requests to the IANA to add them to the registry), but is extensible via the IANA registry to allow additional error codes to be defined in the future.

The R (Retry) flag provides a hint (or suggestion) as to what the receiver may want to do with this annotated error. The mechanism is specifically designed to be extensible, and so implementations may receive EDE codes that it does not understand. The R flag allows implementations to make a decision as to what to do if it receives a response with an unknown code - retry or drop the query. Note that this flag is only a suggestion or hint. Receivers can choose to ignore this hint.

4. Defined Extended DNS Errors

This document defines some initial EDE codes. The mechanism is intended to be extensible, and additional codepoints will be registered in the "Extended DNS Errors" registry. This document provides suggestions for the R flag, but the originating server may ignore these recommendations if it knows better.

4.1. Extended DNS Error Code 100 - DNSSEC Bogus

The resolver attempted to perform DNSSEC validation, but validation ended in the Bogus state. The R flag should not be set.

4.2. Extended DNS Error Code 2 - DNSSEC Indeterminate

The resolver attempted to perform DNSSEC validation, but validation ended in the Indeterminate state.

Usually attached to SERVFAIL messages. The R flag should not be set.

4.3. Extended DNS Error Code 3 - Lame

An authoritative resolver that receives a query (with the RD bit clear) for a domain for which it is not authoritative SHOULD include this EDE code in the REFUSED response.

Implementations should set the R flag in this case (another nameserver might not be lame).

4.4. Extended DNS Error Code 4 - Prohibited

An authoritative or recursive resolver that receives a query from an "unauthorized" client can annotate its REFUSED message with this code. Examples of "unauthorized" clients are recursive queries from IP addresses outside the network, blacklisted IP addresses, etc.

Implementations SHOULD allow operators to define what to set the R flag to in this case.

4.5. Extended DNS Error Code 5 - TooBusy

[Ed: This might be a bad idea. It is intended to allow servers under a DoS (for example a random subdomain attack) to signal to recursive clients that they are being abusive and should back off. This may be a bad idea -- it may "complete the attack", it may be spoofable (by anyone who could also do a MITM style attack), etc.]

A nameserver which is under excessive load (for example, because it is experiencing a DoS) may annotate any answer with this code.

It is RECOMMENDED that implementations set the R flag in this case, but may allow operators to define what to set the R flag to.

[agreed: bad idea -wjh]

5. IANA Considerations

[This section under construction, beware.]

This document defines a new EDNS(0) option, entitled "Extended DNS Error", assigned a value of TBD1 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	Extended DNS Error	TBD	[This document]

Data Tag Name Length Meaning ---- ---- ------ ------- TBD1 FooBar N FooBar server

The IANA is requested to create and maintain the "Extended DNS Error codes" registry. The codepoint space is broken into 3 ranges:

- 1 - 16384: Specification required.
- 16385 - 65000: First Come First Served
- 65000 - 65534: Experimental / Private use

The codepoints 0, 65535 are reserved.

6. Open questions

1 Can this be included in *any* response or only responses to requests that included an EDNS option? Resolvers are supposed to ignore additional. EDNS capable ones are supposed to simply ignore unknown options. I know the spec says you can only include EDNS0 in a response if in a request -- it is time to reevaluate this?

2 Can this be applied to *any* response, or only error responses?

3 Should textual information be allowed as well? What if the only thing allowed is a domain name, e.g to point at where validation began failing?

7. Security Considerations

DNSSEC is being deployed - unfortunately a significant number of clients (~11% according to [GeoffValidation]), when receiving a SERVFAIL from a validating resolver because of a DNSSEC validaion issue simply ask the next (non-validating) resolver in their list, and don't get any of the protections which DNSSEC should provide. This is very similar to a kid asking his mother if he can have another cookie. When the mother says "No, it will ruin your dinner!", going off and asking his (more permissive) father and getting a "Yes, sure, cookie!".

8. Acknowledgements

The authors wish to thank Geoff Huston and Bob Harold, Carlos M. Martinez, Peter DeVries, George Michelson, Mark Andrews, Ondrej Sury, Edward Lewis, Paul Vixie, Shane Kerr. They also vaguely remember discussing this with a number of people over the years, but have forgotten who all they were -- if you were one of them, and are not listed, please let us know and we'll acknowledge you.

I also want to thank the band "Infected Mushroom" for providing a good background soundtrack (and to see if I can get away with this!) Another author would like to thank the band "Mushroom Infectors". This was funny at the time we wrote it, but I cannot remember why...

We would like to especially thank Peter van Dijk, who sent GitHub pull requests.

9. References

9.1. Normative References

[IANA.AS_Numbers]
IANA, "Autonomous System (AS) Numbers", <http://www.iana.org/assignments/as-numbers>.

[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>.

9.2. Informative References

[GeoffValidation]
IANA, "A quick review of DNSSEC Validation in today's Internet", June 2016, <http://www.potaroo.net/presentations/2016-06-27-dnssec.pdf>.

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

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -02 to -03:

- Added David Lawrence -- I somehow missed that in last version.

From -00 to -01;

- Fixed up some of the text, minor clarifications.

Authors' Addresses

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

Email: warren@kumari.net

Evan Hunt
ISC
950 Charter St
Redwood City, CA 94063
US

Email: each@isc.org

Roy Arends
Nominet
UK

Email: TBD

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, VA 95617
US

David C Lawrence
Akamai Technologies
150 Broadway
Cambridge, MA 02142-1054
US

Email: tale@akamai.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 25, 2019

W. Kumari
Google
E. Hunt
ISC
R. Arends
ICANN
W. Hardaker
USC/ISI
D. Lawrence
Oracle + Dyn

September 21, 2018

Extended DNS Errors
draft-ietf-dnsop-extended-error-02

Abstract

This document defines an extensible method to return additional information about the cause of DNS errors. The primary use case is to extend SERVFAIL to provide additional information about the cause of DNS and DNSSEC failures.

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 March 25, 2019.

Table of Contents

1. Introduction and background

There are many reasons that a DNS query may fail, some of them transient, some permanent; some can be resolved by querying another server, some are likely best handled by stopping resolution.

Unfortunately, the error signals that a DNS server can return are very limited, and are not very expressive. This means that applications and resolvers often have to "guess" at what the issue is - e.g the answer was marked REFUSED because of a lame delegation, or because of a lame delegation or because the nameserver is still starting up and loading zones? Is a SERVFAIL a DNSSEC validation issue, or is the nameserver experiencing a bad hair day?

A good example of issues that would benefit by additional error information is an error caused by a DNSSEC validation issue. When a stub resolver queries a DNSSEC bogus name (using a validating resolver), the stub resolver receives only a SERVFAIL in response. Unfortunately, SERVFAIL is used to signal many sorts of DNS errors, and so the stub resolver simply asks the next configured DNS resolver. The result of trying the next resolver is one of two outcomes: either the next resolver also validates, a SERVFAIL is returned again, and the user gets an (largely) incomprehensible error message; or the next resolver is not a validating resolver, and the user is returned a potentially harmful result.

This document specifies a mechanism to extend (or annotate) DNS errors to provide additional information about the cause of the error. This information can be used by the resolver to make a decision regarding whether or not to retry, or by technical users attempting to debug issues.

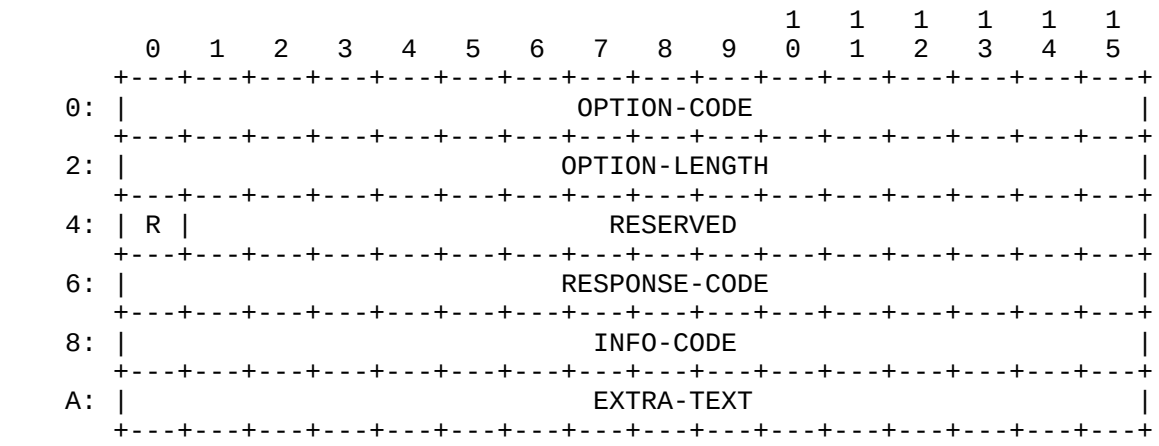
Here is a reference to an "external" (non-RFC / draft) thing: ([IANA.AS_Numbers]). And this is a link to an ID:[I-D.ietf-sidr-iana-objects].

1.1. Requirements notation

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

2. Extended Error EDNS0 option format

This draft uses an EDNS0 ([RFC2671]) option to include extended error (ExtError) information in DNS messages. The option is structured as follows:



- OPTION-CODE, 2 octets (defined in [RFC6891]), for ExtError is TBD.
- OPTION-LENGTH, 2 octets ((defined in [RFC6891]) contains the length of the payload (everything after OPTION-LENGTH) in octets and should be 4.
- RESERVED, 2 octets; the first bit (R) indicates a flag defined in this specification. The remaining bits are reserved for future use, potentially as additional flags.
- RESPONSE-CODE, 2 octets: this SHOULD be a copy of the RCODE from the primary DNS packet. When including multiple extended error EDNS0 records in a response in order to provide additional error information, the RESPONSE-CODE MAY be a different RCODE.
- INFO-CODE, 2 octets.
- A variable length EXTRA-TEXT field holding additional textual information. It may be zero length when no additional textual information is included.

Currently the only defined flag is the R flag.

R - Retry The R (or Retry) flag provides a hint to the receiver that it should retry the query, probably by querying another server. If the R bit is set (1), the sender believes that retrying the query may provide a successful answer next time; if the R bit is clear (0), the sender believes that it should not ask another server.

The remaining bits in the RESERVED field are reserved for future use and MUST be set to 0 by the sender and SHOULD be ignored by the receiver.

INFO-CODE: A code point that, when combined with the RCODE from the DNS packet, serve as a joint-index into the IANA "Extended DNS Errors" registry.

3. Use of the Extended DNS Error option

The Extended DNS Error (EDE) is an EDNS option. It can be included in any error response (SERVFAIL, NXDOMAIN, REFUSED, etc) to a query that includes an EDNS option. This document includes a set of initial codepoints (and requests to the IANA to add them to the registry), but is extensible via the IANA registry to allow additional error and information codes to be defined in the future.

The R (Retry) flag provides a hint (or suggestion) as to what the receiver may want to do with this annotated error. The mechanism is specifically designed to be extensible, and so implementations may receive EDE codes that it does not understand. The R flag allows implementations to make a decision as to what to do if it receives a response with an unknown code - retry or drop the query. Note that this flag is only a suggestion or hint. Receivers can choose to ignore this hint.

The EXTRA-INFO textual field may be zero-length, or may hold additional information useful to network operators.

4. Defined Extended DNS Errors

This document defines some initial EDE codes. The mechanism is intended to be extensible, and additional codepoints will be registered in the "Extended DNS Errors" registry. This document provides suggestions for the R flag, but the originating server may ignore these recommendations if it knows better.

The RESPONSE-CODE and the INFO-CODE from the EDE EDNS option is used to serve as a double index into the "Extended DNS Error codes" IANA registry, the initial values for which are defined in the following sub-sections.

4.1. SERVFAIL(2) extended information codes

4.1.1. Extended DNS Error Code 1 - DNSSEC Bogus

The resolver attempted to perform DNSSEC validation, but validation ended in the Bogus state. The R flag should not be set.

4.1.2. Extended DNS Error Code 2 - DNSSEC Indeterminate

The resolver attempted to perform DNSSEC validation, but validation ended in the Indeterminate state. The R flag should not be set.

4.1.3. Extended DNS Error Code 3 - Signature Expired

The resolver attempted to perform DNSSEC validation, but the signature was expired. The R flag should not be set.

4.1.4. Extended DNS Error Code 4 - Signature Not Yet Valid

The resolver attempted to perform DNSSEC validation, but the signatures received were not yet valid. The R flag should not be set.

4.1.5. Extended DNS Error Code 5 - Unsupported DNSKEY Algorithm

The resolver attempted to perform DNSSEC validation, but a DNSKEY RRSET contained only unknown algorithms. The R flag should not be set.

4.1.6. Extended DNS Error Code 6 - Unsupported DS Algorithm

The resolver attempted to perform DNSSEC validation, but a DS RRSET contained only unknown algorithms. The R flag should not be set.

4.1.7. Extended DNS Error Code 7 - DNSKEY missing

A DS record existed at a parent, but no DNSKEY record could be found for the child. The R flag should not be set.

4.1.8. Extended DNS Error Code 8 - RRSIGs missing

The resolver attempted to perform DNSSEC validation, but no RRSIGs could be found for at least one RRset where RRSIGs were expected.

4.1.9. Extended DNS Error Code 9 - No Zone Key Bit Set

The resolver attempted to perform DNSSEC validation, but no Zone Key Bit was set in a DNSKEY.

4.2. REFUSED(5) extended information codes

4.2.1. Extended DNS Error Code 1 - Lame

An authoritative resolver that receives a query (with the RD bit clear) for a domain for which it is not authoritative SHOULD include this EDE code in the REFUSED response. Implementations should set the R flag in this case (another nameserver might not be lame).

4.2.2. Extended DNS Error Code 2 - Prohibited

An authoritative or recursive resolver that receives a query from an "unauthorized" client can annotate its REFUSED message with this code. Examples of "unauthorized" clients are recursive queries from IP addresses outside the network, blacklisted IP addresses, local policy, etc.

Implementations SHOULD allow operators to define what to set the R flag to in this case.

4.3. NXDOMAIN(3) extended information codes

4.3.1. Extended DNS Error Code 1 - Blocked

The resolver attempted to perfom a DNS query but the domain is blacklisted due to a security policy. The R flag should not be set.

5. IANA Considerations

[This section under construction, beware.]

5.1. new Extended Error Code EDNS Option

This document defines a new EDNS(0) option, entitled "Extended DNS Error", assigned a value of TBD1 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	Extended DNS Error	TBD	[This document]

5.2. new Extended Error Code EDNS Option

This document defines a new double-index IANA registry table, where the first index value is the RCODE value and the second index value is the INFO-CODE from the Extended DNS Error EDNS option defined in

this document. The IANA is requested to create and maintain this "Extended DNS Error codes" registry. The codepoint space for each RCODE index is to be broken into 3 ranges:

- 1 - 16384: Specification required.
- 16385 - 65000: First Come First Served
- 65000 - 65534: Experimental / Private use

The codepoints 0, 65535 are reserved.

A starting table, based on the contents of this document, is as follows:

RCODE	EDE-INFO-CODE	Meaning	Ref
SERVFAIL(2)	DNSSEC_BOGUS(1)	DNSSEC Validation resulted in Bogus	section <xref target="errbogus" />
SERVFAIL(2)	DNSSEC_INDETERMINATE(2)	DNSSEC Validation resulted in Indeterminate	section <xref target="errindeterminate" />

[incomplete]

6. Open questions

1 Can this be included in *any* response or only responses to requests that included an EDNS option? Resolvers are supposed to ignore additional. EDNS capable ones are supposed to simply ignore unknown options. I know the spec says you can only include EDNS0 in a response if in a request -- it is time to reevaluate this?

7. Security Considerations

DNSSEC is being deployed - unfortunately a significant number of clients (~11% according to [GeoffValidation]), when receiving a SERVFAIL from a validating resolver because of a DNSSEC validaion issue simply ask the next (non-validating) resolver in their list, and don’t get any of the protections which DNSSEC should provide. This is very similar to a kid asking his mother if he can have another cookie. When the mother says "No, it will ruin your dinner!", going off and asking his (more permissive) father and getting a "Yes, sure, cookie!".

8. Acknowledgements

The authors wish to thank Geoff Huston and Bob Harold, Carlos M. Martinez, Peter DeVries, George Michelson, Mark Andrews, Ondrej Sury, Edward Lewis, Paul Vixie, Shane Kerr, Loganaden Velvindron. They also vaguely remember discussing this with a number of people over the years, but have forgotten who all they were -- if you were one of them, and are not listed, please let us know and we'll acknowledge you.

I also want to thank the band "Infected Mushroom" for providing a good background soundtrack (and to see if I can get away with this!) Another author would like to thank the band "Mushroom Infectors". This was funny at the time we wrote it, but I cannot remember why...

We would like to especially thank Peter van Dijk, who sent GitHub pull requests.

9. References

9.1. Normative References

[IANA.AS_Numbers]
IANA, "Autonomous System (AS) Numbers", <http://www.iana.org/assignments/as-numbers>.

[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>.

9.2. Informative References

[GeoffValidation]
IANA, "A quick review of DNSSEC Validation in today's Internet", June 2016, <http://www.potaroo.net/presentations/2016-06-27-dnssec.pdf>.

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

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -00 to -01:

- Address comments from IETF meeting.
- document copying the response code
- mention zero length fields are ok
- clarify lookup procedure
- mention that table isn't done

From -03 to -IETF 00:

- Renamed to draft-ietf-dnsop-extended-error

From -02 to -03:

- Added David Lawrence -- I somehow missed that in last version.

From -00 to -01;

- Fixed up some of the text, minor clarifications.

Authors' Addresses

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

Email: warren@kumari.net

Evan Hunt
ISC
950 Charter St
Redwood City, CA 94063
US

Email: each@isc.org

Roy Arends
ICANN

Email: roy.arends@icann.org

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

David C Lawrence
Oracle + Dyn
150 Dow St
Manchester, NH 03101
US

Email: tale@dd.org

DNSOP
Internet-Draft
Updates: 6761 (if approved)
Intended status: Standards Track
Expires: April 27, 2018

M. West
Google, Inc

October 24, 2017

Let ’localhost’ be localhost.
draft-ietf-dnsop-let-localhost-be-localhost-01

Abstract

This document updates RFC6761 with the goal of ensuring that "localhost" can be safely relied upon as a name for the local host’s loopback interface. To that end, stub resolvers are required to resolve localhost names to loopback addresses. Recursive DNS servers are required to return "NXDOMAIN" when queried for localhost names, making non-conformant stub resolvers more likely to fail and produce problem reports that result in updates.

Together, these requirements would allow applications and specifications to join regular users in drawing the common-sense conclusions that "localhost" means "localhost", and doesn’t resolve to somewhere else on the network.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 27, 2018.

Copyright Notice

Table of Contents

1. Introduction

The "127.0.0.0/8" IPv4 address block and "::1/128" IPv6 address block are reserved as loopback addresses. Traffic to this block is assured to remain within a single host, and can not legitimately appear on any network anywhere. This turns out to be a very useful property in a number of circumstances; useful enough to label explicitly and

interoperably as "localhost". [RFC1537] suggests that this special-use top-level domain name has been implicitly mapped to loopback addresses for decades at this point, and that [RFC6761]'s assertion that developers may "assume that IPv4 and IPv6 address queries for localhost names will always resolve to the respective IP loopback address" is well-founded.

Unfortunately, the rest of that latter document's requirements undercut the assumption it suggests. Client software is empowered to send localhost names to DNS servers, and resolvers are empowered to return unexpectedly non-loopback results. This divide between theory and practice has a few impacts:

First, the lack of confidence that "localhost" actually resolves to the loopback interface encourages application developers to hard-code IP addresses like "127.0.0.1" in order to obtain certainty regarding routing. This causes problems in the transition from IPv4 to IPv6 (see problem 8 in [I-D.ietf-sunset4-gapanalysis]).

Second, HTTP user agents sometimes distinguish certain contexts as "secure"-enough to make certain features available. Given the certainty that "127.0.0.1" cannot be maliciously manipulated or monitored, [SECURE-CONTEXTS] treats it as such a context. Since "localhost" might not actually map to the loopback address, that document declines to give it the same treatment. This exclusion has (rightly) surprised some developers, and exacerbates the risks of hard-coded IP addresses by giving developers positive encouragement to use an explicit loopback address rather than a localhost name.

This document updates [RFC6761]'s recommendations regarding "localhost" by requiring that name resolution APIs and libraries themselves return a loopback address when queried for localhost names, bypassing lookup via recursive and authoritative DNS servers entirely.

In addition, recursive and authoritative DNS servers are required to return "NXDOMAIN" for such queries. This increases the likelihood that non-conformant stub resolvers will not go undetected. Note that this does not have the result that such resolvers will fail safe--it just makes it more likely that they will be detected and fixed, since they will fail in the presence of conforming name servers.

These changes are not sufficient to ensure that "localhost" can be assumed to actually refer to an address on the local machine. This document therefore further requires that applications that wish to make that assumption handle the name "localhost" specially.

2. Terminology and 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].

IPv4 loopback addresses are registered in Table 4 of Section 2.2.2 of [RFC6890] as "127.0.0.0/8".

IPv6 loopback addresses are registered in Table 17 of Section 2.2.3 of [RFC6890] as "::1/128".

The domain "localhost.", and any names falling within ".localhost.", are known as "localhost names".

3. The "localhost." Special-Use Domain Name

Localhost names are special insofar as these names do not exist in the DNS, and querying the DNS for them is an error. With that principle in mind, the considerations outlined in [RFC6761] can be answered as follows:

1. Users are free to use localhost names as they would any other domain names. Users may assume that IPv4 and IPv6 address queries for localhost names will always resolve to the respective IP loopback address.

2. Application software MAY recognize localhost names as special, or MAY pass them to name resolution APIs as they would for other domain names.

   If application software wishes to make security decisions based upon the assumption that localhost names resolve to loopback addresses (e.g. if it wishes to ensure that a context meets the requirements laid out in [SECURE-CONTEXTS]), then it MUST directly translate localhost names to a loopback address, and MUST NOT rely upon name resolution APIs to do so.

   Application software MUST NOT use a searchlist to resolve a localhost name. That is, even if DHCP’s domain search option [RFC3397] is used to specify a searchlist of "example.com" for a given network, the name "localhost" will not be resolved as "localhost.example.com." but as "localhost.", and "subdomain.localhost" will not be resolved as "subdomain.localhost.example.com." but as "subdomain.localhost.".

3. Name resolution APIs and libraries MUST recognize localhost names as special, and MUST always return an appropriate IP loopback

address for IPv4 and IPv6 address queries and negative responses for all other query types. Name resolution APIs MUST NOT send queries for localhost names to their configured recursive DNS server(s).

As for application software, name resolution APIs and libraries MUST NOT use a searchlist to resolve a localhost name.

4. (Caching) recursive DNS servers MUST respond to queries for localhost names with NXDOMAIN.

5. Authoritative DNS servers MUST respond to queries for localhost names with NXDOMAIN.

6. DNS server operators SHOULD be aware that the effective RDATA for localhost names is defined by protocol specification and cannot be modified by local configuration.

7. DNS Registries/Registrars MUST NOT grant requests to register localhost names in the normal way to any person or entity. Localhost names are defined by protocol specification and fall outside the set of names available for allocation by registries/ registrars. Attempting to allocate a localhost name as if it were a normal DNS domain name will not work as desired, for reasons 2, 3, 4, and 5 above.

4. IANA Considerations

IANA is requested to update the "localhost." registration in the registry of Special-Use Domain Names [RFC6761] to reference the domain name reservations considerations section of this document.

4.1. Domain Name Reservation Considerations

This document requests that IANA update the "localhost." registration in the registry of Special-Use Domain Names [RFC6761] to reference the domain name reservation considerations defined in Section 3.

4.2. DNSSEC

The ".localhost" TLD is already assigned to IANA, as per [RFC2606], but does not have an entry in the DNSSEC root-zone. This means that the root will return an NXDOMAIN response along with NSEC records constituting a secure denial of existence if queried. That’s consistent with the general principle that localhost names do not exist in the DNS, and the subsequent requirements to return NXDOMAIN that are laid out in Section 3.

5. Security Considerations

5.1. Applications are encouraged to resolve localhost names themselves.

Applications that attempt to use the local resolver to query "localhost" do not fail safely. If an attacker sets up a malicious DNS server which returns a non-loopback address when queried for localhost names, such applications will connect to that remote server assuming it is local. This risk drives the requirement that applications resolve localhost names themselves if they intend to make security decisions based on the assumption that localhost names resolve locally.

There may be cases in which the target runtime environment can be safely assumed to do the right thing with localhost names. In this case, the requirement that the application resolve localhost names on its own may be safe to ignore, but only if all the requirements under point 2 of Section 3 are known to be followed by the resolver that is known to be present in the target environment.

5.2. Non-TLD ’localhost’ labels

Hosts like "localhost.example.com" contain a "localhost" label, but are not affected one way or another by the recommendations in this document. They are not "localhost names", have no resolution guarantees, and should not be given special treatment, either in DNS or in client software.

6. Implementation Considerations

6.1. Non-DNS usage of localhost names

Some application software differentiates between the hostname "localhost" and the IP address "127.0.0.1". MySQL, for example, uses a unix domain socket for the former, and a TCP connection to the loopback address for the latter. The constraints on name resolution APIs above do not preclude this kind of differentiation.

7. References

7.1. 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>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

7.2. Informative References

[I-D.ietf-sunset4-gapanalysis] LIU, W., Xu, W., Zhou, C., Tsou, T., Perreault, S., Fan, P., Gu, R., Xie, C., and Y. Cheng, "Gap Analysis for IPv4 Sunset", draft-ietf-sunset4-gapanalysis-09 (work in progress), August 2017.

[RFC1537] Beertema, P., "Common DNS Data File Configuration Errors", RFC 1537, DOI 10.17487/RFC1537, October 1993, <https://www.rfc-editor.org/info/rfc1537>.

[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>.

[RFC3397] Aboba, B. and S. Cheshire, "Dynamic Host Configuration Protocol (DHCP) Domain Search Option", RFC 3397, DOI 10.17487/RFC3397, November 2002, <https://www.rfc-editor.org/info/rfc3397>.

[SECURE-CONTEXTS] West, M., "Secure Contexts", n.d., <http://w3c.github.io/webappsec-secure-contexts/>.

Appendix A. Changes from RFC 6761

Section 3 updates the requirements in section 6.3 of [RFC6761] in a few substantive ways:

1. Application software and name resolution APIs and libraries are prohibited from using searchlists when resolving localhost names, and encouraged to bypass resolution APIs and libraries altogether if they intend to make security decisions based on the "localhost" name.

2. Name resolution APIs and libraries are required to resolve localhost names to loopback addresses, without sending the query on to caching DNS servers.

3. Caching and authoritative DNS servers are required to respond to resolution requests for localhost names with NXDOMAIN.

Appendix B. Changes in this draft

B.1. draft-ietf-dnsop-let-localhost-be-localhost-01

- Explicit adoption of the principle Wes Hardaker proposed in https://www.ietf.org/mail-archive/web/dnsop/current/msg21039.html , and that Warren Kumari reiterated in https://www.ietf.org/mail-archive/web/dnsop/current/msg21129.html : localhost names do not exist in the DNS, there is no authoritative source for these names, and querying resolvers for them is an error.

- Slight tightening of the admonition against search lists.

- Addressed "localhost" labels in non-localhost names.

B.2. draft-ietf-dnsop-let-localhost-be-localhost-00

- No change since draft-west-let-localhost-be-localhost-06, just renaming the document after DNSOP adopted it.

B.3. draft-west-let-localhost-be-localhost-06

- Incorporated Ted Lemon’s further feedback from https://www.ietf.org/mail-archive/web/dnsop/current/msg20769.html

- Explicitly waffling on DNSSEC.

B.4. draft-west-let-localhost-be-localhost-05

- Updated obsolete references to RFC 5735 and 5156 in favor of [RFC6890].

- Clarify that non-caching recursive DNS servers are also addressed by #4 in Section 3.

- Reformulating the abstract and introduction based on feedback like Ted Lemon’s in https://www.ietf.org/mail-archive/web/dnsop/current/msg20757.html

- Added a request that an insecure delegation for "localhost." be added to the root-zone.

B.5. draft-west-let-localhost-be-localhost-04

- Restructured the draft as a stand-alone document, rather than as set of monkey-patches against [RFC6761].

B.6. draft-west-let-localhost-be-localhost-03

- Explicitly referenced [I-D.ietf-sunset4-gapanalysis].
- Added a prohibition against using searchlists to resolve localhost names.
- Noted that MySQL has special behavior differentiating the connection mechanism used for "localhost" and "127.0.0.1".

B.7. draft-west-let-localhost-be-localhost-02

- Pulled in definitions for IPv4 and IPv6 loopback addresses.

B.8. draft-west-let-localhost-be-localhost-01

- Added a requirement that caching DNS servers MUST generate an immediate negative response.

B.9. draft-west-let-localhost-be-localhost-00

First draft.

Appendix C. Acknowledgements

Ryan Sleevi and Emily Stark informed me about the strange state of localhost name resolution. Erik Nygren poked me to take another look at the set of decisions we made in [SECURE-CONTEXTS] around "localhost."; this document is the result. They, along with Warren Kumari, Ted Lemon, John Levine, Mark Andrews, and many other members of DNSOP offered substantive feedback that markedly improved the quality of this document.

Author’s Address

Mike West
Google, Inc

Email: mkwst@google.com
URI: https://mikewest.org/

DNSOP
Internet-Draft
Updates: 6761 (if approved)
Intended status: Standards Track
Expires: June 21, 2018

M. West
Google, Inc

December 18, 2017

Let ’localhost’ be localhost.
draft-ietf-dnsop-let-localhost-be-localhost-02

Abstract

This document updates the treatment of the special-use domain name "localhost" as specified in RFC6761, Section 6.3, with the goal of ensuring that it can be safely relied upon as a name for the local host’s loopback interface. To that end, stub resolvers are required to resolve localhost names to loopback addresses. Recursive DNS servers are required to return "NXDOMAIN" when queried for localhost names, making non-conformant stub resolvers more likely to fail and produce problem reports that result in updates.

Together, these requirements would allow applications and specifications to join regular users in drawing the common-sense conclusions that "localhost" means "localhost", and doesn’t resolve to somewhere else on the network.

Status of This Memo

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

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

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

This Internet-Draft will expire on June 21, 2018.

Table of Contents

1. Introduction

The "127.0.0.0/8" IPv4 address block and "::1/128" IPv6 address block are reserved as loopback addresses. Traffic to these blocks is assured to remain within a single host, and can not legitimately appear on any network anywhere. This turns out to be a very useful

property in a number of circumstances; useful enough to label explicitly and interoperably as "localhost". [RFC1537] suggests that this special-use top-level domain name has been implicitly mapped to loopback addresses for decades at this point, and that [RFC6761]'s assertion that developers may "assume that IPv4 and IPv6 address queries for localhost names will always resolve to the respective IP loopback address" is well-founded.

Unfortunately, the rest of that latter document's requirements undercut the assumption it suggests. Client software is empowered to send localhost names to DNS servers, and resolvers are empowered to return unexpectedly non-loopback results. This divide between theory and practice has a few impacts:

First, the lack of confidence that "localhost" actually resolves to the loopback interface encourages application developers to hard-code IP addresses like "127.0.0.1" in order to obtain certainty regarding routing. This causes problems in the transition from IPv4 to IPv6 (see problem 8 in [I-D.ietf-sunset4-gapanalysis]).

Second, HTTP user agents sometimes distinguish certain contexts as "secure"-enough to make certain features available. Given the certainty that "127.0.0.1" cannot be maliciously manipulated or monitored, [SECURE-CONTEXTS] treats it as such a context. Since "localhost" might not actually map to the loopback address, that document declines to give it the same treatment. This exclusion has (rightly) surprised some developers, and exacerbates the risks of hard-coded IP addresses by giving developers positive encouragement to use an explicit loopback address rather than a localhost name.

This document updates [RFC6761]'s recommendations regarding "localhost" by requiring that name resolution APIs and libraries themselves return a loopback address when queried for localhost names, bypassing lookup via recursive and authoritative DNS servers entirely.

In addition, recursive and authoritative DNS servers are required to return "NXDOMAIN" for such queries. This increases the likelihood that non-conformant stub resolvers will not go undetected. Note that this does not have the result that such resolvers will fail safe--it just makes it more likely that they will be detected and fixed, since they will fail in the presence of conforming name servers.

These changes are not sufficient to ensure that "localhost" can be assumed to actually refer to an address on the local machine. This document therefore further requires that applications that wish to make that assumption handle the name "localhost" specially.

2. Terminology and 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].

IPv4 loopback addresses are registered in Table 4 of Section 2.2.2 of [RFC6890] as "127.0.0.0/8".

IPv6 loopback addresses are registered in Table 17 of Section 2.2.3 of [RFC6890] as "::1/128".

The domain "localhost.", and any names falling within ".localhost.", are known as "localhost names".

3. The "localhost." Special-Use Domain Name

Localhost names are special insofar as these names do not exist in the DNS, and querying the DNS for them is an error. With that principle in mind, the considerations outlined in [RFC6761] can be answered as follows:

1. Users are free to use localhost names as they would any other domain names. Users may assume that IPv4 and IPv6 address queries for localhost names will always resolve to the respective IP loopback address.

2. Application software MAY recognize localhost names as special, or MAY pass them to name resolution APIs as they would for other domain names.

   If application software wishes to make security decisions based upon the assumption that localhost names resolve to loopback addresses (e.g. if it wishes to ensure that a context meets the requirements laid out in [SECURE-CONTEXTS]), then it MUST directly translate localhost names to a loopback address, and MUST NOT rely upon name resolution APIs to do so.

   Application software MUST NOT use a searchlist to resolve a localhost name. That is, even if DHCP’s domain search option [RFC3397] is used to specify a searchlist of "example.com" for a given network, the name "localhost" will not be resolved as "localhost.example.com." but as "localhost.", and "subdomain.localhost" will not be resolved as "subdomain.localhost.example.com." but as "subdomain.localhost.".

3. Name resolution APIs and libraries MUST recognize localhost names as special, and MUST always return an appropriate IP loopback

address for IPv4 and IPv6 address queries and negative responses for all other query types. Name resolution APIs MUST NOT send queries for localhost names to their configured recursive DNS server(s).

As for application software, name resolution APIs and libraries MUST NOT use a searchlist to resolve a localhost name.

4. (Caching) recursive DNS servers MUST respond to queries for localhost names with NXDOMAIN.

5. Authoritative DNS servers MUST respond to queries for localhost names with NXDOMAIN.

6. DNS server operators SHOULD be aware that the effective RDATA for localhost names is defined by protocol specification and cannot be modified by local configuration.

7. DNS Registries/Registrars MUST NOT grant requests to register localhost names in the normal way to any person or entity. Localhost names are defined by protocol specification and fall outside the set of names available for allocation by registries/ registrars. Attempting to allocate a localhost name as if it were a normal DNS domain name will not work as desired, for reasons 2, 3, 4, and 5 above.

4. IANA Considerations

4.1. Domain Name Reservation Considerations

This document requests that IANA updates the "localhost." registration in the registry of Special-Use Domain Names [RFC6761] to reference this document rather than [RFC6761].

Considerations for this reservation are detailed in Section 3.

4.2. DNSSEC

The ".localhost" TLD is already assigned to IANA, as per [RFC2606], but does not have an entry in the root-zone. This means that the root will return an NXDOMAIN response along with NSEC records constituting a secure denial of existence if queried. That’s consistent with the general principle that localhost names do not exist in the DNS, and the subsequent requirements to return NXDOMAIN that are laid out in Section 3.

5. Security Considerations

5.1. Applications are encouraged to resolve localhost names themselves.

Applications that attempt to use the local resolver to query "localhost" do not fail safely. If an attacker sets up a malicious DNS server which returns a non-loopback address when queried for localhost names, such applications will connect to that remote server assuming it is local. This risk drives the requirement that applications resolve localhost names themselves if they intend to make security decisions based on the assumption that localhost names resolve locally.

There may be cases in which the target runtime environment can be safely assumed to do the right thing with localhost names. In this case, the requirement that the application resolve localhost names on its own may be safe to ignore, but only if all the requirements under point 2 of Section 3 are known to be followed by the resolver that is known to be present in the target environment.

5.2. ’localhost’ labels in subdomains

Hosts like "localhost.example.com" and "subdomain.localhost.example.com" contain a "localhost" label, but are not themselves localhost names, as they do not fall within "localhost.". Therefore, they are not directly affected by the recommendations in this document. They have no resolution guarantees one way or another, and should not be given special treatment, either in DNS or in client software.

Note, however, that the admonition against searchlist usage could affect their resolution in practice, as discussed in Section 3. For example, even with a searchlist of "example.com" in place for a given network, the name "localhost" will not be resolved as "localhost.example.com." but as "localhost.", and "subdomain.localhost" will not be resolved as "subdomain.localhost.example.com." but as "subdomain.localhost.".

6. Implementation Considerations

6.1. Non-DNS usage of localhost names

Some application software differentiates between the hostname "localhost" and the IP address "127.0.0.1". MySQL, for example, uses a unix domain socket for the former, and a TCP connection to the loopback address for the latter. The constraints on name resolution APIs above do not preclude this kind of differentiation.

7. References

7.1. 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>.

[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>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

7.2. Informative References

[I-D.ietf-sunset4-gapanalysis] LIU, W., Xu, W., Zhou, C., Tsou, T., Perreault, S., Fan, P., Gu, R., Xie, C., and Y. Cheng, "Gap Analysis for IPv4 Sunset", draft-ietf-sunset4-gapanalysis-09 (work in progress), August 2017.

[RFC1537] Beertema, P., "Common DNS Data File Configuration Errors", RFC 1537, DOI 10.17487/RFC1537, October 1993, <https://www.rfc-editor.org/info/rfc1537>.

[RFC3397] Aboba, B. and S. Cheshire, "Dynamic Host Configuration Protocol (DHCP) Domain Search Option", RFC 3397, DOI 10.17487/RFC3397, November 2002, <https://www.rfc-editor.org/info/rfc3397>.

[SECURE-CONTEXTS] West, M., "Secure Contexts", n.d., <http://w3c.github.io/webappsec-secure-contexts/>.

Appendix A. Changes from RFC 6761

Section 3 updates the requirements in section 6.3 of [RFC6761] in a few substantive ways:

1. Application software and name resolution APIs and libraries are prohibited from using searchlists when resolving localhost names, and encouraged to bypass resolution APIs and libraries altogether if they intend to make security decisions based on the "localhost" name.

2. Name resolution APIs and libraries are required to resolve localhost names to loopback addresses, without sending the query on to caching DNS servers.

3. Caching and authoritative DNS servers are required to respond to resolution requests for localhost names with NXDOMAIN.

Appendix B. Changes in this draft

B.1. draft-ietf-dnsop-let-localhost-be-localhost-02

- Based on some feedback from Suzanne Woolf, this draft:
  - Clarified the abstract (https://github.com/mikewest/internetdrafts/commit/837b89f35e08e98b0e02df87032c4ccc19cd06eb)
  - Addressed nits in the "IANA considerations" section (https://github.com/mikewest/internetdrafts/commit/d65d4fbaec6afbbae70496ffb98dfb60e8d3e2eb)
  - Reworded the "Non-TLD localhost" section (https://github.com/mikewest/internetdrafts/commit/44b1d7d4cfcb65aab3c46ff1c436a75a2fb3403f)
  - Made the reference to [RFC2606] normative (https://github.com/mikewest/internetdrafts/commit/cd94988a966b93d2239de03d54513031a5823c0a)

B.2. draft-ietf-dnsop-let-localhost-be-localhost-01

- Explicit adoption of the principle Wes Hardaker proposed in https://www.ietf.org/mail-archive/web/dnsop/current/msg21039.html , and that Warren Kumari reiterated in https://www.ietf.org/mail-archive/web/dnsop/current/msg21129.html : localhost names do not exist in the DNS, there is no authoritative source for these names, and querying resolvers for them is an error.
- Slight tightening of the admonition against search lists.
- Addressed "localhost" labels in non-localhost names.

B.3. draft-ietf-dnsop-let-localhost-be-localhost-00

- No change since draft-west-let-localhost-be-localhost-06, just renaming the document after DNSOP adopted it.

B.4. draft-west-let-localhost-be-localhost-06

- Incorporated Ted Lemon's further feedback from https://www.ietf.org/mail-archive/web/dnsop/current/msg20769.html
- Explicitly waffling on DNSSEC.

B.5. draft-west-let-localhost-be-localhost-05

- Updated obsolete references to RFC 5735 and 5156 in favor of [RFC6890].
- Clarify that non-caching recursive DNS servers are also addressed by #4 in Section 3.
- Reformulating the abstract and introduction based on feedback like Ted Lemon's in https://www.ietf.org/mail-archive/web/dnsop/current/msg20757.html
- Added a request that an insecure delegation for "localhost." be added to the root-zone.

B.6. draft-west-let-localhost-be-localhost-04

- Restructured the draft as a stand-alone document, rather than as set of monkey-patches against [RFC6761].

B.7. draft-west-let-localhost-be-localhost-03

- Explicitly referenced [I-D.ietf-sunset4-gapanalysis].
- Added a prohibition against using searchlists to resolve localhost names.
- Noted that MySQL has special behavior differentiating the connection mechanism used for "localhost" and "127.0.0.1".

B.8. draft-west-let-localhost-be-localhost-02

- Pulled in definitions for IPv4 and IPv6 loopback addresses.

B.9. draft-west-let-localhost-be-localhost-01

- Added a requirement that caching DNS servers MUST generate an immediate negative response.

B.10. draft-west-let-localhost-be-localhost-00

First draft.

Appendix C. Acknowledgements

Ryan Sleevi and Emily Stark informed me about the strange state of localhost name resolution. Erik Nygren poked me to take another look at the set of decisions we made in [SECURE-CONTEXTS] around "localhost."; this document is the result. They, along with Warren Kumari, Ted Lemon, John Levine, Mark Andrews, and many other members of DNSOP offered substantive feedback that markedly improved the quality of this document.

Author’s Address

Mike West
Google, Inc

Email: mkwst@google.com
URI: https://mikewest.org/

dnsop
Internet-Draft
Updates: 7583 (if approved)
Intended status: Standards Track
Expires: April 21, 2018

W. Hardaker
USC/ISI
W. Kumari
Google

October 18, 2017

Security Considerations for RFC5011 Publishers
draft-ietf-dnsop-rfc5011-security-considerations-07

Abstract

This document extends the RFC5011 rollover strategy with timing advice that must be followed in order to maintain security. Specifically, this document describes the math behind the minimum time-length that a DNS zone publisher must wait before signing exclusively with recently added DNSKEYs. It contains much math and complicated equations, but the summary is that the key rollover / revocation time is much longer than intuition would suggest. If you are not both publishing a DNSSEC trust anchor, and using RFC5011 to update that trust anchor, you probably don’t need to read this document.

This document also describes the minimum time-length that a DNS zone publisher must wait after publishing a revoked DNSKEY before assuming that all active RFC5011 resolvers should have seen the revocation-marked key and removed it from their list of trust anchors.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2018.

Table of Contents

1. Introduction

[RFC5011] defines a mechanism by which DNSSEC validators can update their list of trust anchors when they've seen a new key published in a zone. However, RFC5011 [intentionally] provides no guidance to the publishers of DNSKEYs about how long they must wait before switching to exclusively using recently published keys for signing records, or how long they must wait before ceasing publication of a revoked key. Because of this lack of guidance, zone publishers may derive incorrect assumptions about safe usage of the RFC5011 DNSKEY advertising, rolling and revocation process. This document describes the minimum security requirements from a publisher's point of view and is intended to complement the guidance offered in RFC5011 (which is written to provide timing guidance solely to a Validating Resolver's point of view).

1.1. Document History and Motivation

To verify this lack of understanding is wide-spread, the authors reached out to 5 DNSSEC experts to ask them how long they thought they must wait before signing a zone exclusively with a new KSK [RFC4033] that was being introduced according to the 5011 process. All 5 experts answered with an insecure value, and we determined that this lack of operational guidance is causing security concerns today and wrote this companion document to RFC5011. We hope that this document will rectify this understanding and provide better guidance to zone publishers that wish to make use of the RFC5011 rollover process.

1.2. Safely Rolling the Root Zone's KSK in 2017/2018

One important note about ICANN's [currently upcoming] 2017/2018 KSK rollover plan for the root zone: the timing values chosen for rolling the KSK in the root zone appear completely safe, and are not affected by the timing concerns introduced by this draft

1.3. Requirements notation

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

2. Background

The RFC5011 process describes a process by which a RFC5011 Validating Resolver may accept a newly published KSK as a trust anchor for validating future DNSSEC signed records. It also describes the process for publicly revoking a published KSK. This document

augments that information with additional constraints, from the DNSKEY publication and revocation's points of view. Note that this document does not define any other operational guidance or recommendations about the RFC5011 process and restricts itself to solely the security and operational ramifications of switching to exclusively using recently added keys or removing a revoked keys too soon.

Failure of a DNSKEY publisher to follow the minimum recommendations associated with this draft will result in potential denial-of-service attack opportunities against validating resolvers. Failure of a DNSKEY publisher to publish a revoked key for a long enough period of time may result in RFC5011 Validating Resolvers leaving that key in their trust anchor storage beyond the key's expected lifetime.

3. Terminology

Trust Anchor Publisher The entity responsible for publishing a DNSKEY that can be used as a trust anchor.

Zone Signer The owner of a zone intending to publish a new Key-Signing-Key (KSK) that will become a trust anchor by validators following the RFC5011 process.

RFC5011 Validating Resolver A DNSSEC Validating Resolver that is using the RFC5011 processes to track and update trust anchors. Sometimes referred to as a "RFC5011 Resolver"

Attacker An entity intent on foiling the RFC5011 Validator's ability to successfully adopt the Zone Signer's new DNSKEY as a new trust anchor or to prevent the RFC5011 Validator from removing an old DNSKEY from its list of trust anchors.

sigExpirationTime The amount of time remaining before the latest RRSIG's Signature Expiration time is reached. Note that for organizations pre-creating signatures this time may be fairly lengthy unless they can be significantly assured their signatures can not be replayed at a later date. sigExpirationTime will fundamentally be the RRSIG's Signature Expiration time minus the RRSIG's Signature Inception time when the signature is created.

Also see Section 2 of [RFC4033] and [RFC7719] for additional terminology.

4. Timing Associated with RFC5011 Processing

These sections define a high-level overview of [RFC5011] processing. These steps are not sufficient for proper RFC5011 implementation, but provide enough background for the reader to follow the discussion in this document. Readers need to fully understand [RFC5011] as well to fully comprehend the importance of this document.

4.1. Timing Associated with Publication

RFC5011's process of safely publishing a new key and then making use of that key falls into a number of high-level steps to be performed by the Trust Anchor Publisher. This document discusses the following scenario, which the principle way RFC5011 is currently being used (even though Section 6 of RFC5011 suggests having a stand-by key available):

1. Publish a new DNSKEY in the zone, but continue to sign the zone with the old one.

2. Wait a period of time.

3. Begin to exclusively use recently published DNSKEYs to sign the appropriate resource records.

This document discusses step 2 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". Section 5 describes an attack based on this (common) erroneous belief, which can result in a denial of service attack against the zone.

4.2. Timing Associated with Revocation

RFC5011's process of advertising that an old key is to be revoked from RFC5011 validating resolvers falls into a number of high-level steps:

1. Set the revoke bit on the DNSKEY to be revoked.

2. Sign the revoked DNSKEY with itself.

3. Wait a period of time.

4. Remove the revoked key from the zone.

This document discusses step 3 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". This document describes

an attack based on this (common) erroneous belief, which results in a revoked DNSKEY potentially remaining as a trust anchor in a RFC5011 validating resolver long past its expected usage.

5. Denial of Service Attack Considerations

If an attacker is able to provide a RFC5011 Validating Resolver with past responses, such as when it is in-path or able to perform any number of cache poisoning attacks, the attacker may be able to leave compliant RFC5011-Validating Resolvers without an appropriate DNSKEY trust anchor. This scenario will remain until an administrator manually fixes the situation.

The time-line below illustrates this situation.

5.1. Enumerated Attack Example

The following example settings are used in the example scenario within this section:

TTL (all records) 1 day

sigExpirationTime 10 days

Zone resigned every 1 day

Given these settings, the sequence of events in Section 5.1.1 depicts how a Trust Anchor Publisher that waits for only the RFC5011 hold time timer length of 30 days subjects its users to a potential Denial of Service attack. The timing schedule listed below is based on a Trust Anchor Publisher publishing a new Key Signing Key (KSK), with the intent that it will later become a trust anchor. We label this publication time as "T+0". All numbers in this sequence refer to days before and after this initial publication event. Thus, T-1 is the day before the introduction of the new key, and T+15 is the 15th day after the key was introduced into the fictitious zone being discussed.

In this dialog, we consider two keys within the example zone:

K_old An older KSK and Trust Anchor being replaced.

K_new A new KSK being transitioned into active use and expected to become a Trust Anchor via the RFC5011 process.

5.1.1. Attack Timing Breakdown

The steps shows an attack that foils the adoption of a new DNSKEY by a 5011 Validating Resolver when the Trust Anchor Publisher that starts signing and publishing with the new DNSKEY too quickly.

T-1 The K_old based RRSIGs are being published by the Zone Signer. [It may also be signing ZSKs as well, but they are not relevant to this event so we will not talk further about them; we are only considering the RRSIGs that cover the DNSKEYs in this document.] The Attacker queries for, retrieves and caches this DNSKEY set and corresponding RRSIG signatures. Note that for simplicity we assume the signer is not pre-signing and creating "valid in the future" signature sets that may be stolen and replayed even later.

T+0 The Zone Signer adds K_new to their zone and signs the zone's key set with K_old. The RFC5011 Validator (later to be under attack) retrieves this new key set and corresponding RRSIGs and notices the publication of K_new. The RFC5011 Validator starts the (30-day) hold-down timer for K_new. [Note that in a more real-world scenario there will likely be a further delay between the point where the Zone Signer publishes a new RRSIG and the RFC5011 Validator notices its publication; though not shown in this example, this delay is accounted for in the final solution below]

T+5 The RFC5011 Validator queries for the zone's keyset per the RFC5011 Active Refresh schedule, discussed in Section 2.3 of RFC5011. Instead of receiving the intended published keyset, the Attacker successfully replays the keyset and associated signatures recorded at T-1. Because the signature lifetime is 10 days (in this example), the replayed signature and keyset is accepted as valid (being only 6 days old, which is less than sigExpirationTime) and the RFC5011 Validator cancels the hold-down timer for K_new, per the RFC5011 algorithm.

T+10 The RFC5011 Validator queries for the zone's keyset and discovers a signed keyset that includes K_new (again), and is signed by K_old. Note: the attacker is unable to replay the records cached at T-1, because they have now expired. Thus at T+10, the RFC5011 Validator starts (anew) the hold-timer for K_new.

T+11 through T+29 The RFC5011 Validator continues checking the zone's key set at the prescribed regular intervals. During this period, the attacker can no longer replay traffic to their benefit.

T+30 The Zone Signer knows that this is the first time at which some validators might accept K_new as a new trust anchor, since the hold-down timer of a RFC5011 Validator not under attack that had queried and retrieved K_new at T+0 would now have reached 30 days. However, the hold-down timer of our attacked RFC5011 Validator is only at 20 days.

T+35 The Zone Signer (mistakenly) believes that all validators following the Active Refresh schedule (Section 2.3 of RFC5011) should have accepted K_new as a the new trust anchor (since the hold down time (30 days) + the query interval [which is just 1/2 the signature validity period in this example] would have passed). However, the hold-down timer of our attacked RFC5011 Validator is only at 25 days (T+35 minus T+10); thus the RFC5011 won't consider it a valid trust anchor addition yet, as the required 30 days have not yet elapsed.

T+36 The Zone Signer, believing K_new is safe to use, switches their active signing KSK to K_new and publishes a new RRSIG, signed with K_new, covering the DNSKEY set. Non-attacked RFC5011 validators, with a hold-down timer of at least 30 days, would have accepted K_new into their set of trusted keys. But, because our attacked RFC5011 Validator now has a hold-down timer for K_new of only 26 days, it failed to accept K_new as a trust anchor. Since K_old is no longer being used to sign the zone's DNSKEYs, all the DNSKEY records from the zone will be treated as invalid. Subsequently, all of the records in the DNS tree below the zone's apex will be deemed invalid by DNSSEC.

6. Minimum RFC5011 Timing Requirements

6.1. Timing Requirements For Adding a New KSK

Given the attack description in Section 5, the correct minimum length of time required for the Zone Signer to wait after publishing K_new but before exclusively using it and newer keys is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTime
              + activeRefresh
              + activeRefreshOffset
              + safetyMargin
```

6.1.1. addHoldDownTime

The addHoldDownTime is defined in Section 2.4.1 of [RFC5011] as:

> The add hold-down time is 30 days or the expiration time of the original TTL of the first trust point DNSKEY RRSet that contained the new key, whichever is greater. This ensures that at least two validated DNSKEY RRSets that contain the new key MUST be seen by the resolver prior to the key's acceptance.

6.1.2. sigExpirationTime

sigExpirationTime is defined in Section 3.

6.1.3. activeRefresh

activeRefresh time is defined by RFC5011 by

> A resolver that has been configured for an automatic update of keys from a particular trust point MUST query that trust point (e.g., do a lookup for the DNSKEY RRSet and related RRSIG records) no less often than the lesser of 15 days, half the original TTL for the DNSKEY RRSet, or half the RRSIG expiration interval and no more often than once per hour.

This translates to:

```
activeRefresh = MAX(1 hour,
                    MIN(sigExpirationTime / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days)
                    )
```

6.1.4. activeRefreshOffset

The activeRefreshOffset term must be added for situations where the activeRefresh value is not a factor of the addHoldDownTime. Specifically, activeRefreshOffset will be "addHoldDownTime % activeRefresh", where % is the mathematical mod operator (calculating the remainder in a division problem). This will frequently be zero, but could be nearly as large as activeRefresh itself. For simplicity, setting the activeRefreshOffset to the activeRefresh value itself is always safe.

6.1.5. safetyMargin

The safetyMargin is an extra period of time to account for caching, network delays, etc. A suggested operational value for this is 2 * MAX(TTL of all records) unless the TTLs are shorter than an hour, at which point they start affecting the calculations in the MIN() clause

in the activeRefresh timer in Section 6.1.3. Thus, we suggest a safetyMargin of at least:

```
safetyMargin = MAX (1.5 hours, 2 * MAX(TTL of all records))
```

RFC5011 also discusses a retryTime value for failed queries. Our equation cannot take into account undeterministic failure situations, so it might be wise to extend the safetyMargin by some factor of retryTime, which is defined in RFC5011 as:

```
retryTime = MAX (1 hour,
                 MIN (1 day,
                      .1 * TTL of K_old DNSKEY RRset,
                      .1 * sigExpirationTime))
```

6.1.6. Fully expanded equation

The full expanded equation, with activeRefreshOffset set to activeRefresh for simplicity, is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTime
              + 2 * MAX(1 hour,
                        MIN(sigExpirationTime / 2,
                            MAX(TTL of K_old DNSKEY RRSet) / 2,
                            15 days)
                        )
              + (addHoldDownTime % activeRefresh)
              + MAX(1.5 hours, 2 * MAX(TTL of all records))
```

6.1.7. Timing Constraint Summary

The important timing constraint introduced by this memo relates to the last point at which a validating resolver may have received a replayed original DNSKEY set, containing K_old and not K_new. The next query of the RFC5011 validator at which K_new will be seen without the potential for a replay attack will occur after the publication time plus sigExpirationTime. Thus, the latest time that a RFC5011 Validator may begin their hold down timer is an "Active Refresh" period after the last point that an attacker can replay the K_old DNSKEY set. The worst case scenario of this attack is if the attacker can replay K_old seconds before the (DNSKEY RRSIG Signature Validity) field of the last K_old only RRSIG.

6.1.8. Additional Considerations

Note: our notion of addWaitTime is called "Itrp" in Section 3.3.4.1 of [RFC7583]. The equation for Itrp in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Itrp equation in RFC7583 also does not include the 2*TTL safety margin, though that is an operational consideration and not necessarily as critical.

6.1.8.1. Example Results

For the parameters listed in Section 5.1, the activeRefreshOffset is 0, since 30 days is evenly divisible by activeRefresh (1/2 day), and our resulting addWaitTime is:

```
addWaitTime = 30
              + 10
              + 1 / 2
              + 2 * (1)        (days)

addWaitTime = 42.5             (days)
```

This addWaitTime of 42.5 days is 12.5 days longer than just the hold down timer.

6.2. Timing Requirements For Revoking an Old KSK

It is important to note that this issue affects not just the publication of new DNSKEYs intended to be used as trust anchors, but also the length of time required to continuously publish a DNSKEY with the revoke bit set. Both of these publication timing requirements are affected by the attacks described in this document, but with revocation the key is revoked immediately and the addHoldDown timer does not apply. Thus the minimum amount of time that a Trust Anchor Publisher must wait before removing a revoked key from publication is:

```
remWaitTime = sigExpirationTime
              + MAX(1 hour,
                    MIN((sigExpirationTime) / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days),
                    1 hour)
              + 2 * MAX(TTL of all records)
```

Note that the activeRefreshOffset time does not apply to this equation.

Note that our notion of remWaitTime is called "Irev" in Section 3.3.4.2 of [RFC7583]. The equation for Irev in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Irev equation in RFC7583 also does not include the 2*TTL safety margin, though that is an operational consideration and not necessarily as critical.

Note also that adding retryTime intervals to the remWaitTime may be wise, just as it was for addWaitTime in Section 6.

6.2.1. Example Results

For the parameters listed in Section 5.1, our example:

```
  remwaitTime = 10
                + 1 / 2
                + 2 * (1)        (days)

  remwaitTime = 12.5             (days)
```

Note that for the values in this example produce a length shorter than the recommended 30 days in RFC5011's section 6.6, step 3. Other values of sigExpirationTime and the original TTL of the K_old DNSKEY RRSet, however, can produce values longer than 30 days.

Note that because revocation happens immediately, an attacker has a much harder job tricking a RFC5011 Validator into leaving a trust anchor in place, as the attacker must successfully replay the old data for every query a RFC5011 Validator sends, not just one.

7. IANA Considerations

This document contains no IANA considerations.

8. Operational Considerations

A companion document to RFC5011 was expected to be published that describes the best operational practice considerations from the perspective of a zone publisher and Trust Anchor Publisher. However, this companion document has yet to be published. The authors of this document hope that it will at some point in the future, as RFC5011 timing can be tricky as we have shown, and a BCP is clearly warranted. This document is intended only to fill a single operational void which, when left misunderstood, can result in serious security ramifications. This document does not attempt to document any other missing operational guidance for zone publishers.

9. Security Considerations

This document, is solely about the security considerations with respect to the Trust Anchor Publisher of RFC5011 trust anchors / DNSKEYs. Thus the entire document is a discussion of Security Considerations when adding or removing DNSKEYs from trust anchor storage using the RFC5011 process.

For simplicity, this document assumes that the Trust Anchor Publisher will use a consistent RRSIG validity period. Trust Anchor Publishers that vary the length of RRSIG validity periods will need to adjust the sigExpirationTime value accordingly so that the equations in Section 6 and Section 6.2 use a value that coincides with the last time a replay of older RRSIGs will no longer succeed.

10. Acknowledgements

The authors would like to especially thank to Michael StJohns for his help and advice and the care and thought he put into RFC5011 itself. We would also like to thank Bob Harold, Shane Kerr, Matthijs Mekking, Duane Wessels, Petr Petr Spacek, Ed Lewis, and the dnsop working group who have assisted with this document.

11. Normative References

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

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

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

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

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

Appendix A. Real World Example: The 2017 Root KSK Key Roll

In 2017, ICANN expects to (or has, depending on when you're reading this) roll the key signing key (KSK) for the root zone. The relevant parameters associated with the root zone at the time of this writing is as follows:

```
addHoldDownTime:                    30 days
Old DNSKEY sigExpirationTime:       21 days
Old DNSKEY TTL:                      2 days
```

Thus, sticking this information into the equation in Section Section 6 yields (in days):

```
addWaitTime = 30
              + (21)
              + MAX(MIN((21) / 2,
                        MAX(2 / 2,
                        15 days)),
                    1 hour)
              + 2 * MAX(2)

addWaitTime = 30 + 21 + MAX(MIN(11.5, 1, 15)), 1 hour) + 4

addWaitTime = 30 + 21 + 1 + 4

addWaitTime = 56 days
```

Note that we use a activeRefreshOffset of 0, since 30 days is evenly divisible by activeRefresh (1 day).

Thus, ICANN should wait a minimum of 56 days before switching to the newly published KSK (and 26 days before removing the old revoked key once it is published as revoked). ICANN's current plans are to wait 70 days before using the new KEY and 69 days before removing the old, revoked key. Thus, their current rollover plans are sufficiently secure from the attack discussed in this memo.

Authors' Addresses

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

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

Email: warren@kumari.net

dnsop
Internet-Draft
Updates: 7583 (if approved)
Intended status: Informational
Expires: January 17, 2019

W. Hardaker
USC/ISI
W. Kumari
Google

July 16, 2018

Security Considerations for RFC5011 Publishers
draft-ietf-dnsop-rfc5011-security-considerations-13

Abstract

This document extends the RFC5011 rollover strategy with timing advice that must be followed by the publisher in order to maintain security. Specifically, this document describes the math behind the minimum time-length that a DNS zone publisher must wait before signing exclusively with recently added DNSKEYs. This document also describes the minimum time-length that a DNS zone publisher must wait after publishing a revoked DNSKEY before assuming that all active RFC5011 resolvers should have seen the revocation-marked key and removed it from their list of trust anchors.

This document contains much math and complicated equations, but the summary is that the key rollover / revocation time is much longer than intuition would suggest. This document updates RFC7583 by adding an additional delays (sigExpirationTime and timingSafetyMargin).

If you are not both publishing a DNSSEC DNSKEY, and using RFC5011 to advertise this DNSKEY as a new Secure Entry Point key for use as a trust anchor, you probably don’t need to read this document.

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 January 17, 2019.

Table of Contents

1. Introduction

[RFC5011] defines a mechanism by which DNSSEC validators can update their list of trust anchors when they've seen a new key published in a zone or revoke a properly marked key from a trust anchor list. However, RFC5011 [intentionally] provides no guidance to the publishers of DNSKEYs about how long they must wait before switching to exclusively using recently published keys for signing records, or how long they must wait before ceasing publication of a revoked key. Because of this lack of guidance, zone publishers may arrive at incorrect assumptions about safe usage of the RFC5011 DNSKEY advertising, rolling and revocation process. This document describes the minimum security requirements from a publisher's point of view and is intended to complement the guidance offered in RFC5011 (which is written to provide timing guidance solely to a Validating Resolver's point of view).

To explain the RFC5011 security analysis in this document better, Section 5 first describes an attack on a zone publisher. Then in Section 6.1 we break down each of the timing components that will be later used to define timing requirements for adding keys in Section 6.2 and revoking keys in Section 6.3.

1.1. Document History and Motivation

To confirm that this lack of understanding is wide-spread, the authors reached out to 5 DNSSEC experts to ask them how long they thought they must wait before signing a zone exclusively with a new KSK [RFC4033] that was being introduced according to the 5011 process. All 5 experts answered with an insecure value, and we determined that this lack of understanding might cause security concerns in deployment. We hope that this companion document to RFC5011 will rectify this and provide better guidance to zone publishers who wish to make use of the RFC5011 rollover process.

1.2. Safely Rolling the Root Zone's KSK in 2017/2018

One important note about ICANN's (currently in process) 2017/2018 KSK rollover plan for the root zone: the timing values chosen for rolling the KSK in the root zone appear completely safe, and are not affected by the timing concerns discussed in this draft.

1.3. Requirements notation

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

2. Background

RFC5011 describes a process by which an RFC5011 Resolver may accept a newly published KSK as a trust anchor for validating future DNSSEC signed records. It also describes the process for publicly revoking a published KSK. This document augments that information with additional constraints, from the SEP publisher's points of view. Note that this document does not define any other operational guidance or recommendations about the RFC5011 process and restricts itself solely to the security and operational ramifications of prematurely switching to exclusively using recently added keys or removing revoked keys.

Failure of a DNSKEY publisher to follow the minimum recommendations associated with this draft can result in potential denial-of-service attack opportunities against validating resolvers. Failure of a DNSKEY publisher to publish a revoked key for a long enough period of time may result in RFC5011 Resolvers leaving that key in their trust anchor storage beyond the key's expected lifetime.

3. Terminology

SEP Publisher The entity responsible for publishing a DNSKEY (with the Secure Entry Point (SEP) bit set) that can be used as a trust anchor.

Zone Signer The owner of a zone intending to publish a new Key-Signing-Key (KSK) that may become a trust anchor for validators following the RFC5011 process.

RFC5011 Resolver A DNSSEC Resolver that is using the RFC5011 processes to track and update trust anchors.

Attacker An entity intent on foiling the RFC5011 Resolver's ability to successfully adopt the Zone Signer's new DNSKEY as a new trust

anchor or to prevent the RFC5011 Resolver from removing an old DNSKEY from its list of trust anchors.

sigExpirationTime The amount of time between the DNSKEY RRSIG's Signature Inception field and the Signature Expiration field.

Also see Section 2 of [RFC4033] and [RFC7719] for additional terminology.

4. Timing Associated with RFC5011 Processing

These subsections below give a high-level overview of [RFC5011] processing. This description is not sufficient for fully understanding RFC5011, but provide enough background for the reader to follow the discussion in this document. Readers need to fully understand [RFC5011] as well to fully comprehend the content and importance of this document.

4.1. Timing Associated with Publication

RFC5011's process of safely publishing a new DNSKEY and then assuming RFC5011 Resolvers have adopted it for trust can be broken down into a number of high-level steps to be performed by the SEP Publisher. This document discusses the following scenario, which the principal way RFC5011 is currently being used (even though Section 6 of RFC5011 suggests having a stand-by key available):

1. Publish a new DNSKEY in a zone, but continue to sign the zone with the old one.

2. Wait a period of time.

3. Begin to exclusively use recently published DNSKEYs to sign the appropriate resource records.

This document discusses the time required to wait during step 2 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". Section 5 describes an attack based on this (common) erroneous belief, which can result in a denial of service attack against the zone.

4.2. Timing Associated with Revocation

RFC5011's process of advertising that an old key is to be revoked from RFC5011 Resolvers falls into a number of high-level steps:

1. Set the revoke bit on the DNSKEY to be revoked.

2. Sign the revoked DNSKEY with itself.

3. Wait a period of time.

4. Remove the revoked key from the zone.

This document discusses the time required to wait in step 3 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". This document describes an attack based on this (common) erroneous belief, which results in a revoked DNSKEY potentially remaining as a trust anchor in a RFC5011 Resolver long past its expected usage.

5. Denial of Service Attack Walkthrough

This section serves as an illustrative example of the problem being discussed in this document. Note that in order to keep the example simple enough to understand, some simplifications were made (such as by not creating a set of pre-signed RRSIGs and by not using values that result in the addHoldDownTime not being evenly divisible by the activeRefresh value); the mathematical formulas in Section 6 are, however, complete.

If an attacker is able to provide a RFC5011 Resolver with past responses, such as when it is on-path or able to perform any number of cache poisoning attacks, the attacker may be able to leave compliant RFC5011 Resolvers without an appropriate DNSKEY trust anchor. This scenario will remain until an administrator manually fixes the situation.

The time-line below illustrates an example of this situation.

5.1. Enumerated Attack Example

The following settings are used in the example scenario within this section:

TTL (all records) 1 day

sigExpirationTime 10 days

Zone resigned every 1 day

Given these settings, the sequence of events in Section 5.1.1 depicts how a SEP Publisher that waits for only the RFC5011 hold time timer length of 30 days subjects its users to a potential Denial of Service attack. The timeline below is based on a SEP Publisher publishing a new Key Signing Key (KSK), with the intent that it will later be used

as a trust anchor. We label this publication time as "T+0". All numbers in this timeline refer to days before and after this initial publication event. Thus, T-1 is the day before the introduction of the new key, and T+15 is the 15th day after the key was introduced into the example zone being discussed.

In this exposition, we consider two keys within the example zone:

K_old: An older KSK and Trust Anchor being replaced.

K_new: A new KSK being transitioned into active use and expected to become a Trust Anchor via the RFC5011 automated trust anchor update process.

5.1.1. Attack Timing Breakdown

Below we examine an attack that foils the adoption of a new DNSKEY by a 5011 Resolver when the SEP Publisher that starts signing and publishing with the new DNSKEY too quickly.

T-1 The K_old based RRSIGs are being published by the Zone Signer. [It may also be signing ZSKs as well, but they are not relevant to this event so we will not talk further about them; we are only considering the RRSIGs that cover the DNSKEYs in this document.] The Attacker queries for, retrieves and caches this DNSKEY set and corresponding RRSIG signatures.

T+0 The Zone Signer adds K_new to their zone and signs the zone’s key set with K_old. The RFC5011 Resolver (later to be under attack) retrieves this new key set and corresponding RRSIGs and notices the publication of K_new. The RFC5011 Resolver starts the (30-day) hold-down timer for K_new. [Note that in a more real-world scenario there will likely be a further delay between the point where the Zone Signer publishes a new RRSIG and the RFC5011 Resolver notices its publication; though not shown in this example, this delay is accounted for in the equation in Section 6 below]

T+5 The RFC5011 Resolver queries for the zone’s keyset per the RFC5011 Active Refresh schedule, discussed in Section 2.3 of RFC5011. Instead of receiving the intended published keyset, the Attacker successfully replays the keyset and associated signatures recorded at T-1 to the victim RFC5011 Resolver. Because the signature lifetime is 10 days (in this example), the replayed signature and keyset is accepted as valid (being only 6 days old, which is less than sigExpirationTime) and the RFC5011 Resolver cancels the (30-day) hold-down timer for K_new, per the RFC5011 algorithm.

T+10 The RFC5011 Resolver queries for the zone's keyset and discovers a signed keyset that includes K_new (again), and is signed by K_old. Note: the attacker is unable to replay the records cached at T-1, because the signatures have now expired. Thus at T+10, the RFC5011 Resolver starts (anew) the hold-timer for K_new.

T+11 through T+29 The RFC5011 Resolver continues checking the zone's key set at the prescribed regular intervals. During this period, the attacker can no longer replay traffic to their benefit.

T+30 The Zone Signer knows that this is the first time at which some validators might accept K_new as a new trust anchor, since the hold-down timer of a RFC5011 Resolver not under attack that had queried and retrieved K_new at T+0 would now have reached 30 days. However, the hold-down timer of our attacked RFC5011 Resolver is only at 20 days.

T+35 The Zone Signer (mistakenly) believes that all validators following the Active Refresh schedule (Section 2.3 of RFC5011) should have accepted K_new as a the new trust anchor (since the hold down time (30 days) + the query interval [which is just 1/2 the signature validity period in this example] would have passed). However, the hold-down timer of our attacked RFC5011 Resolver is only at 25 days (T+35 minus T+10); thus the RFC5011 Resolver won't consider it a valid trust anchor addition yet, as the required 30 days have not yet elapsed.

T+36 The Zone Signer, believing K_new is safe to use, switches their active signing KSK to K_new and publishes a new RRSIG, signed with (only) K_new, covering the DNSKEY set. Non-attacked RFC5011 validators, with a hold-down timer of at least 30 days, would have accepted K_new into their set of trusted keys. But, because our attacked RFC5011 Resolver now has a hold-down timer for K_new of only 26 days, it failed to ever accept K_new as a trust anchor. Since K_old is no longer being used to sign the zone's DNSKEYs, all the DNSKEY records from the zone will be treated as invalid. Subsequently, all of the records in the DNS tree below the zone's apex will be deemed invalid by DNSSEC.

6. Minimum RFC5011 Timing Requirements

This section defines the minimum timing requirements for making exclusive use of newly added DNSKEYs and timing requirements for ceasing the publication of DNSKEYs to be revoked. We break our timing solution requirements into two primary components: the mathematically-based security analysis of the RFC5011 publication

process itself, and an extension of this that takes operational realities into account that further affect the recommended timings.

First, we define the component terms used in all equations in Section 6.1.

6.1. Equation Components

6.1.1. addHoldDownTime

The addHoldDownTime is defined in Section 2.4.1 of [RFC5011] as:

> The add hold-down time is 30 days or the expiration time of the original TTL of the first trust point DNSKEY RRSet that contained the new key, whichever is greater. This ensures that at least two validated DNSKEY RRSets that contain the new key MUST be seen by the resolver prior to the key's acceptance.

6.1.2. lastSigExpirationTime

The latest value (i.e. the future most date and time) of any RRSig Signature Expiration field covering any DNSKEY RRSet containing only the old trust anchor(s) that are being superseded. Note that for organizations pre-creating signatures this time may be fairly far in the future unless they can be significantly assured that none of their pre-generated signatures can be replayed at a later date.

6.1.3. sigExpirationTime

The amount of time between the DNSKEY RRSIG's Signature Inception field and the Signature Expiration field.

6.1.4. sigExpirationTimeRemaining

sigExpirationTimeRemaining is defined in Section 3.

6.1.5. activeRefresh

activeRefresh time is defined by RFC5011 by

> A resolver that has been configured for an automatic update of keys from a particular trust point MUST query that trust point (e.g., do a lookup for the DNSKEY RRSet and related RRSIG records) no less often than the lesser of 15 days, half the original TTL for the DNSKEY RRSet, or half the RRSIG expiration interval and no more often than once per hour.

This translates to:

```
activeRefresh = MAX(1 hour,
                    MIN(sigExpirationTime / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days)
                    )
```

6.1.6. timingSafetyMargin

Mentally, it is easy to assume that the period of time required for SEP publishers to wait after making changes to SEP marked DNSKEY sets will be entirely based on the length of the addHoldDownTime. Unfortunately, analysis shows that both the design of the RFC5011 protocol an the operational realities in deploying it require waiting and additional period of time longer. In subsections Section 6.1.6.1 to Section 6.1.6.3 below, we discuss three sources of additional delay. In the end, we will pick the largest of these delays as the minimum additional time that the SEP Publisher must wait in our final timingSafetyMargin value, which we define in Section 6.1.6.4.

6.1.6.1. activeRefreshOffset

A security analysis of the timing associated with the query rate of RFC5011 Resolvers shows that it may not perfectly align with the addHoldDownTime when the addHoldDownTime is not evenly divisible by the activeRefresh time. Consider the example of a zone with an activeRefresh period of 7 days. If an associated RFC5011 Resolver started it’s holdDown timer just after the SEP published a new DNSKEY (at time T+0), the resolver would send checking queries at T+7, T+14, T+21 and T+28 Days and will finally accept it at T+35 days, which is 5 days longer than the 30-day addHoldDownTime.

The activeRefreshOffset term defines this time difference and becomes:

```
activeRefreshOffset = addHoldDownTime % activeRefresh
```

The % symbol denotes the mathematical mod operator (calculating the remainder in a division problem). This will frequently be zero, but can be nearly as large as activeRefresh itself.

6.1.6.2. clockskewDriftMargin

Even small clock drifts can have negative impacts upon the timing of the RFC5011 Resolver’s measurements. Consider the simplest case where the RFC5011 Resolver’s clock shifts over time to be 2 seconds

slower near the end of the RFC5011 Resolver's addHoldDownTime period. I.E., if the RFC5011 Resolver first noticed a new DNSKEY at:

firstSeen = sigExpirationTime + activeRefresh + 1 second

The effect of 2 second clock drift between the SEP Publisher and the RFC5011 Resolver may result in the RFC5011 Resolver querying again at:

justBefore = sigExpirationTime + addHoldDownTime + activeRefresh + 1 second - 2 seconds

which becomes:

justBefore = sigExpirationTime + addHoldDownTime + activeRefresh - 1 second

The net effect is the addHoldDownTime will not have been reached from the perspective of the RFC5011 Resolver, but it will have been reached from the perspective of the SEP Publisher. The net effect is it may take one additional activeRefresh period longer for this RFC5011 Resolver to accept the new key (at sigExpirationTime + addHoldDownTime + 2 * activeRefresh - 1 second).

We note that even the smallest clockskew errors can require waiting an additional activeRefresh period, and thus define the clockskewDriftMargin as:

clockskewDriftMargin = activeRefresh

6.1.6.3. retryDriftMargin

Drift associated with a lost transmission and an accompanying re-transmission (see Section 2.3 of [RFC5011]) will cause RFC5011 Resolvers to also change the timing associated with query times such that it becomes impossible to predict, from the perspective of the SEP Publisher, when the conclusive measurement query will arrive. Similarly, any software that restarts/reboots without saving next-query timing state may also commence with a new random starting time. Thus, an additional activeRefresh is needed to handle both these cases as well.

retryDriftMargin = activeRefresh

Note that we account for additional time associated with cumulative multiple retries, especially under high-loss conditions, in Section 6.1.6.4.

6.1.6.4. timingSafetyMargin Value

The activeRefreshOffset, clockskewDriftMargin, and retryDriftMargin parameters all deal with additional wait-periods that must be accounted for after analyzing what conditions the client will take longer than expected to make its last query while waiting for the addHoldDownTime period to pass. But these values may be merged into a single term by waiting the longest of any of them. We define timingSafetyMargin as this "worst case" value:

```
timingSafetyMargin = MAX(activeRefreshOffset,
                         clockskewDriftMargin,
                         retryDriftMargin)

timingSafetyMargin = MAX(addWaitTime % activeRefresh,
                         activeRefresh,
                         activeRefresh)

timingSafetyMargin = activeRefresh
```

6.1.7. retrySafetyMargin

The retrySafetyMargin is an extra period of time to account for caching, network delays, dropped packets, and other operational concerns otherwise beyond the scope of this document. The value operators should chose is highly dependent on the deployment situation associated with their zone. Note that no value of a retrySafetyMargin can protect against resolvers that are "down". Nonetheless, we do offer the following as one method considering reasonable values to select from.

The following list of variables need to be considered when selecting an appropriate retrySafetyMargin value:

successRate: A likely success rate for client queries and retries

numResolvers: The number of client RFC5011 Resolvers

Note that RFC5011 defines retryTime as:

> If the query fails, the resolver MUST repeat the query until satisfied no more often than once an hour and no less often than the lesser of 1 day, 10% of the original TTL, or 10% of the original expiration interval. That is,
> retryTime = MAX (1 hour, MIN (1 day, .1 * origTTL,
> .1 * expireInterval)).

With the successRate and numResolvers values selected and the definition of retryTime from RFC5011, one method for determining how many retryTime intervals to wait in order to reduce the set of resolvers that have not accepted the new trust anchor to 0 is thus:

$$x = (1/(1 - successRate))$$

$$retryCountWait = Log\_base\_x(numResolvers)$$

To reduce the need for readers to pull out a scientific calculator, we offer the following lookup table based on successRate and numResolvers:

retryCountWait lookup table

		Number of client RFC5011 Resolvers (numResolvers)				
		10,000	100,000	1,000,000	10,000,000	100,000,000
	0.01	917	1146	1375	1604	1833
Probability	0.05	180	225	270	315	360
of Success	0.10	88	110	132	153	175
Per Retry	0.15	57	71	86	100	114
Interval	0.25	33	41	49	57	65
(successRate)	0.50	14	17	20	24	27
	0.90	4	5	6	7	8
	0.95	4	4	5	6	7
	0.99	2	3	3	4	4
	0.999	2	2	2	3	3

Finally, a suggested value of retrySafetyMargin can then be this retryCountWait number multiplied by the retryTime from RFC5011:

$$retrySafetyMargin = retryCountWait * retryTime$$

6.2. Timing Requirements For Adding a New KSK

Given the defined parameters and analysis from Section 6.1, we can now create a method for calculating the amount of time to wait until it is safe to start signing exclusively with a new DNSKEY (especially useful for writing code involving sleep based timers) in Section 6.2.1, and define a method for calculating a wall-clock value after which it is safe to start signing exclusively with a new DNSKEY (especially useful for writing code based on clock-based event triggers) in Section 6.2.2.

6.2.1. Wait Timer Based Calculation

Given the attack description in Section 5, the correct minimum length of time required for the Zone Signer to wait after publishing K_new but before exclusively using it and newer keys is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTimeRemaining
              + activeRefresh
              + timingSafetyMargin
              + retrySafetyMargin
```

6.2.1.1. Fully expanded equation

Given the equation components defined in Section 6.1, the full expanded equation is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTimeRemaining
              + 2 * MAX(1 hour,
                    MIN(sigExpirationTime / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days)
                    )
              + retrySafetyMargin
```

6.2.2. Wall-Clock Based Calculation

The equations in Section 6.2.1 are defined based upon how long to wait from a particular moment in time. An alternative, but equivalent, method is to calculate the date and time before which it is unsafe to use a key for signing. This calculation thus becomes:

```
addWallClockTime = lastSigExpirationTime
                   + addHoldDownTime
                   + activeRefresh
                   + timingSafetyMargin
                   + retrySafetyMargin
```

where lastSigExpirationTime is the latest value of any sigExpirationTime for which RRSIGs were created that could potentially be replayed. Fully expanded, this becomes:

```
addWallClockTime = lastSigExpirationTime
                   + addHoldDownTime
                   + 2 * MAX(1 hour,
                             MIN(sigExpirationTime / 2,
                                 MAX(TTL of K_old DNSKEY RRSet) / 2,
                                 15 days)
                             )
                   + retrySafetyMargin
```

6.2.3. Timing Constraint Summary

The important timing constraint introduced by this memo relates to the last point at which a RFC5011 Resolver may have received a replayed original DNSKEY set, containing K_old and not K_new. The next query of the RFC5011 validator at which K_new will be seen without the potential for a replay attack will occur after the old DNSKEY RRSIG's Signature Expriation Time. Thus, the latest time that a RFC5011 Validator may begin their hold down timer is an "Active Refresh" period after the last point that an attacker can replay the K_old DNSKEY set. The worst case scenario of this attack is if the attacker can replay K_old just seconds before the (DNSKEY RRSIG Signature Validity) field of the last K_old only RRSIG.

6.2.4. Additional Considerations for RFC7583

Note: our notion of addWaitTime is called "Itrp" in Section 3.3.4.1 of [RFC7583]. The equation for Itrp in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Itrp equation in RFC7583 also does not include the 2*TTL safety margin, though that is an operational consideration.

6.2.5. Example Scenario Calculations

For the parameters listed in Section 5.1, our resulting addWaitTime is:

```
addWaitTime = 30
              + 10
              + 1 / 2
              + 1 / 2          (days)

addWaitTime = 43               (days)
```

This addWaitTime of 42.5 days is 12.5 days longer than just the hold down timer, even with the needed retrySafetyMargin value being left out (which we exclude due to the lack of necessary operational parameters).

6.3. Timing Requirements For Revoking an Old KSK

This issue affects not just the publication of new DNSKEYs intended to be used as trust anchors, but also the length of time required to continuously publish a DNSKEY with the revoke bit set.

Section 6.2.1 defines a method for calculating the amount of time operators need to wait until it is safe to cease publishing a DNSKEY (especially useful for writing code involving sleep based timers), and Section 6.2.2 defines a method for calculating a minimal wall-clock value after which it is safe to cease publishing a DNSKEY (especially useful for writing code based on clock-based event triggers).

6.3.1. Wait Timer Based Calculation

Both of these publication timing requirements are affected by the attacks described in this document, but with revocation the key is revoked immediately and the addHoldDown timer does not apply. Thus the minimum amount of time that a SEP Publisher must wait before removing a revoked key from publication is:

```
remWaitTime = sigExpirationTimeRemaining
              + activeRefresh
              + timingSafetyMargin
              + retrySafetyMargin

remWaitTime = sigExpirationTimeRemaining
              + MAX(1 hour,
                    MIN((sigExpirationTime) / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days))
              + activeRefresh
              + retrySafetyMargin
```

Note also that adding retryTime intervals to the remWaitTime may be wise, just as it was for addWaitTime in Section 6.

6.3.2. Wall-Clock Based Calculation

Like before, the above equations are defined based upon how long to wait from a particular moment in time. An alternative, but equivalent, method is to calculate the date and time before which it is unsafe to cease publishing a revoked key. This calculation thus becomes:

```
remWallClockTime = lastSigExpirationTime
                 + activeRefresh
                 + timingSafetyMargin
                 + retrySafetyMargin

remWallClockTime = lastSigExpirationTime
                 + MAX(1 hour,
                       MIN((sigExpirationTime) / 2,
                           MAX(TTL of K_old DNSKEY RRSet) / 2,
                           15 days))
                 + timingSafetyMargin
                 + retrySafetyMargin
```

where lastSigExpirationTime is the latest value of any sigExpirationTime for which RRSIGs were created that could potentially be replayed. Fully expanded, this becomes:

6.3.3. Additional Considerations for RFC7583

Note that our notion of remWaitTime is called "Irev" in Section 3.3.4.2 of [RFC7583]. The equation for Irev in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Irev equation in RFC7583 also does not include a safety margin, though that is an operational consideration.

6.3.4. Example Scenario Calculations

For the parameters listed in Section 5.1, our example:

```
remwaitTime = 10
            + 1 / 2          (days)

remwaitTime = 10.5           (days)
```

Note that for the values in this example produce a length shorter than the recommended 30 days in RFC5011's section 6.6, step 3. Other values of sigExpirationTime and the original TTL of the K_old DNSKEY RRSet, however, can produce values longer than 30 days.

Note that because revocation happens immediately, an attacker has a much harder job tricking a RFC5011 Resolver into leaving a trust anchor in place, as the attacker must successfully replay the old data for every query a RFC5011 Resolver sends, not just one.

7. IANA Considerations

This document contains no IANA considerations.

8. Operational Considerations

A companion document to RFC5011 was expected to be published that describes the best operational practice considerations from the perspective of a zone publisher and SEP Publisher. However, this companion document has yet to be published. The authors of this document hope that it will at some point in the future, as RFC5011 timing can be tricky as we have shown, and a BCP is clearly warranted. This document is intended only to fill a single operational void which, when left misunderstood, can result in serious security ramifications. This document does not attempt to document any other missing operational guidance for zone publishers.

9. Security Considerations

This document, is solely about the security considerations with respect to the SEP Publisher’s ability to advertise new DNSKEYs via the RFC5011 automated trust anchor update process. Thus the entire document is a discussion of Security Considerations when adding or removing DNSKEYs from trust anchor storage using the RFC5011 process.

For simplicity, this document assumes that the SEP Publisher will use a consistent RRSIG validity period. SEP Publishers that vary the length of RRSIG validity periods will need to adjust the sigExpirationTime value accordingly so that the equations in Section 6 and Section 6.3 use a value that coincides with the last time a replay of older RRSIGs will no longer succeed.

10. Acknowledgements

The authors would like to especially thank to Michael StJohns for his help and advice and the care and thought he put into RFC5011 itself and his continued reviews and suggestions for this document. He also designed the suggested math behind the suggested retrySafetyMargin values in Section 6.1.7.

We would also like to thank Bob Harold, Shane Kerr, Matthijs Mekking, Duane Wessels, Petr Petr Spacek, Ed Lewis, Viktor Dukhovni, and the dnsop working group who have assisted with this document.

11. 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>.

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

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

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

Appendix A. Real World Example: The 2017 Root KSK Key Roll

In 2017 and 2018, ICANN expects to (or has, depending on when you’re reading this) roll the key signing key (KSK) for the root zone. The relevant parameters associated with the root zone at the time of this writing is as follows:

```
addHoldDownTime:                    30 days
Old DNSKEY sigExpirationTime:       21 days
Old DNSKEY TTL:                      2 days
```

Thus, sticking this information into the equation in Section Section 6 yields (in days from publication time):

```
addWaitTime = 30
              + 21
              + MAX(1 hour,
                    MIN(21 / 2,     # activeRefresh
                        MAX(2) / 2,
                        15 days),
                    )
              + activeRefresh

addWaitTime = 30 + 21 + 1 + 1

addWaitTime = 53 days
```

Also note that we exclude the retrySafetyMargin value, which is calculated based on the expected client deployment size.

Thus, ICANN must wait a minimum of 52 days before switching to the newly published KSK (and 26 days before removing the old revoked key once it is published as revoked). ICANN's current plans involve waiting over 3 months before using the new KEY and 69 days before removing the old, revoked key. Thus, their current rollover plans are sufficiently secure from the attack discussed in this memo.

Authors' Addresses

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

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

Email: warren@kumari.net

Network Working Group
Internet-Draft
Obsoletes: 7719 (if approved)
Updates: 2308 (if approved)
Intended status: Best Current Practice
Expires: April 21, 2018

P. Hoffman
ICANN
A. Sullivan
Oracle
K. Fujiwara
JPRS

October 18, 2017

DNS Terminology
draft-ietf-dnsop-terminology-bis-07

Abstract

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

This document will be the successor to RFC 7719, and thus will obsolete RFC 7719. It will also update RFC 2308.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2018.

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) is a simple query-response protocol whose messages in both directions have the same format. (See Section 2 for a fuller definition.) The protocol and message format are defined in [RFC1034] and [RFC1035]. These RFCs defined some terms, but later documents defined others. Some of the terms from [RFC1034] and [RFC1035] now have somewhat different meanings than they did in 1987.

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

Most of the definitions here are the consensus definition of the DNS community -- both protocol developers and operators. Some of the definitions differ from earlier RFCs, and those differences are

noted. In this document, where the consensus definition is the same as the one in an RFC, that RFC is quoted. Where the consensus definition has changed somewhat, the RFC is mentioned but the new stand-alone definition is given. See Appendix A for a list of the definitions that this document updates.

It is important to note that, during the development of this document, it became clear that some DNS-related terms are interpreted quite differently by different DNS experts. Further, some terms that are defined in early DNS RFCs now have definitions that are generally agreed to, but that are different from the original definitions. Therefore, this document is a substantial revision to [RFC7719].

The terms are organized loosely by topic. Some definitions are for new terms for things that are commonly talked about in the DNS community but that never had terms defined for them.

Other organizations sometimes define DNS-related terms their own way. For example, the W3C defines "domain" at https://specs.webplatform.org/url/webspecs/develop/. The Root Server System Advisory Committee (RSSAC) has a good lexicon [RSSAC026].

Note that there is no single consistent definition of "the DNS". It can be considered to be some combination of the following: a commonly used naming scheme for objects on the Internet; a distributed database representing the names and certain properties of these objects; an architecture providing distributed maintenance, resilience, and loose coherency for this database; and a simple query-response protocol (as mentioned below) implementing this architecture. Section 2 defines "global DNS" and "private DNS" as a way to deal with these differing definitions.

Capitalization in DNS terms is often inconsistent among RFCs and various DNS practitioners. The capitalization used in this document is a best guess at current practices, and is not meant to indicate that other capitalization styles are wrong or archaic. In some cases, multiple styles of capitalization are used for the same term due to quoting from different RFCs.

2. Names

Naming system: A naming system associates names with data. Naming systems have many significant facets that help differentiate them. Some commonly-identified facets include:

* Composition of names

* Format of names

* Administration of names

* Types of data that can be associated with names

* Types of metadata for names

* Protocol for getting data from a name

* Context for resolving a name

Note that this list is a small subset of facets that people have identified over time for naming systems, and the IETF has yet to agree on a good set of facets that can be used to compare naming systems. For example, other facets might include "protocol to update data in a name", "privacy of names", and "privacy of data associated with names", but those are not as well-defined as the ones listed above. The list here is chosen because it helps describe the DNS and naming systems similar to the DNS.

Domain name: An ordered list of one or more labels.

Note that this is a definition independent of the DNS RFCs, and the definition here also applies to systems other than the DNS. [RFC1034] defines the "domain name space" using mathematical trees and their nodes in graph theory, and the definition in [RFC1034] has the same practical result as the definition here. Using graph theory, a domain name is a list of labels identifying a portion along one edge of a directed acyclic graph. A domain name can be relative to other parts of the tree, or it can be fully qualified (in which case, it begins at the common root of the graph).

Also note that different IETF and non-IETF documents have used the term "domain name" in many different ways. It is common for earlier documents to use "domain name" to mean "names that match the syntax in [RFC1035]", but possibly with additional rules such as "and are, or will be, resolvable in the global DNS" or "but only using the presentation format".

Label: An ordered list of zero or more octets and which makes up a portion of a domain name. Using graph theory, a label identifies one node in a portion of the graph of all possible domain names.

Global DNS: Using the short set of facets listed in "Naming system", the global DNS can be defined as follows. Most of the rules here come from [RFC1034] and [RFC1035], although the term "global DNS" has not been defined before now.

Composition of names -- A name in the global DNS has one or more labels. The length of each label is between 0 and 63 octets inclusive. In a fully-qualified domain name, the first label in the ordered list is 0 octets long; it is the only label whose length may be 0 octets, and it is called the "root" or "root label". A domain name in the global DNS has a maximum total length of 255 octets in the wire format; the root represents one octet for this calculation.

Format of names -- Names in the global DNS are domain names. There are three formats: wire format, presentation format, and common display.

The basic wire format for names in the global DNS is a list of labels ordered by decreasing distance from the root, with the root label last. Each label is preceded by a length octet. [RFC1035] also defines a compression scheme that modifies this format.

The presentation format for names in the global DNS is a list of labels ordered by decreasing distance from the root, encoded as ASCII, with a "." character between each label. In presentation format, a fully-qualified domain name includes the root label and the associated separator dot. For example, in presentation format, a fully-qualified domain name with two non-root labels is always shown as "example.tld." instead of "example.tld". [RFC1035] defines a method for showing octets that do not display in ASCII.

The common display format is used in applications and free text. It is the same as the presentation format, but showing the root label and the "." before it is optional and is rarely done. For example, in common display format, a fully-qualified domain name with two non-root labels is usually shown as "example.tld" instead of "example.tld.". Names in the common display format are normally written such that the directionality of the writing system presents labels by decreasing distance from the root (so, in both English and C the first label in the ordered list is right-most; but in Arabic it may be left-most, depending on local conventions).

Administration of names -- Administration is specified by delegation (see the definition of to "delegation" in Section 6). Policies for administration of the root zone in the global DNS are determined by the names operational community, which convenes itself in the Internet Corporation for Assigned Names and Numbers (ICANN). The names operational community selects the IANA Functions Operator for the global DNS root zone. At the time this document is published, that operator is Public Technical

Identifiers (PTI). The name servers that serve the root zone are provided by independent root operators. Other zones in the global DNS have their own policies for administration.

Types of data that can be associated with names -- A name can have zero or more resource records associated with it. There are numerous types of resource records with unique data structures defined in many different RFCs and in the IANA registry at [IANA_Resource_Registry].

Types of metadata for names -- Any name that is published in the DNS appears as a set of resource records (see the definition of "RRset" in Section 4). Some names do not themselves have data associated with them in the DNS, but "appear" in the DNS anyway because they form part of a longer name that does have data associated with it (see the defintion of "empty non-terminals" in Section 6).

Protocol for getting data from a name -- The protocol described in [RFC1035].

Context for resolving a name -- The global DNS root zone distributed by PTI.

Private DNS: Names that use the protocol described in [RFC1035] but that do not rely on the global DNS root zone, or names that are otherwise not generally available on the Internet but are using the protocol described in [RFC1035]. A system can use both the global DNS and one or more private DNS systems; for example, see "Split DNS" in Section 7.

Note that domain names that do not appear in the DNS, and that are intended never to be looked up using the DNS protocol, are not part of the global DNS or a private DNS even though they are domain names.

Locally served DNS zone: A locally served DNS zone is a special case of private DNS. Names are resolved using the DNS protocol in a local context. [RFC6303] defines subdomains of IN-ADDR.ARPA that are locally served zones. Resolution of names through locally served zones may result in ambiguous results. For example, the same name may resolve to different results in different locally served DNS zone contexts. The context through which a locally served zone may be explicit, for example, as defined in [RFC6303], or implicit, as defined by local DNS administration and not known to the resolution client.

Fully qualified domain name (FQDN): This is often just a clear way of saying the same thing as "domain name of a node", as outlined above. However, the term is ambiguous. Strictly speaking, a fully qualified domain name would include every label, including the zero-length label of the root: such a name would be written "www.example.net." (note the terminating dot). But because every name eventually shares the common root, names are often written relative to the root (such as "www.example.net") and are still called "fully qualified". This term first appeared in [RFC0819]. In this document, names are often written relative to the root.

The need for the term "fully qualified domain name" comes from the existence of partially qualified domain names, which are names where one or more of the earliest labels in the ordered list are omitted (for example, "www"). Such relative names are understood only by context.

Host name: This term and its equivalent, "hostname", have been widely used but are not defined in [RFC1034], [RFC1035], [RFC1123], or [RFC2181]. The DNS was originally deployed into the Host Tables environment as outlined in [RFC0952], and it is likely that the term followed informally from the definition there. Over time, the definition seems to have shifted. "Host name" is often meant to be a domain name that follows the rules in Section 3.5 of [RFC1034], the "preferred name syntax". Note that any label in a domain name can contain any octet value; hostnames are generally considered to be domain names where every label follows the rules in the "preferred name syntax", with the amendment that labels can start with ASCII digits (this amendment comes from Section 2.1 of [RFC1123]).

People also sometimes use the term hostname to refer to just the first label of an FQDN, such as "printer" in "printer.admin.example.com". (Sometimes this is formalized in configuration in operating systems.) In addition, people sometimes use this term to describe any name that refers to a machine, and those might include labels that do not conform to the "preferred name syntax".

TLD: A Top-Level Domain, meaning a zone that is one layer below the root, such as "com" or "jp". There is nothing special, from the point of view of the DNS, about TLDs. Most of them are also delegation-centric zones, and there are significant policy issues around their operation. TLDs are often divided into sub-groups such as Country Code Top-Level Domains (ccTLDs), Generic Top-Level Domains (gTLDs), and others; the division is a matter of policy, and beyond the scope of this document.

IDN: The common abbreviation for "Internationalized Domain Name". The IDNA protocol is the standard mechanism for handling domain names with non-ASCII characters in applications in the DNS. The current standard, normally called "IDNA2008", is defined in [RFC5890], [RFC5891], [RFC5892], [RFC5893], and [RFC5894]. These documents define many IDN-specific terms such as "LDH label", "A-label", and "U-label". [RFC6365] defines more terms that relate to internationalization (some of which relate to IDNs), and [RFC6055] has a much more extensive discussion of IDNs, including some new terminology.

Subdomain: "A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name." (Quoted from [RFC1034], Section 3.1). For example, in the host name "nnn.mmm.example.com", both "mmm.example.com" and "nnn.mmm.example.com" are subdomains of "example.com".

Alias: The owner of a CNAME resource record, or a subdomain of the owner of a DNAME resource record. See also "canonical name".

Canonical name: A CNAME resource record "identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR." (Quoted from [RFC1034], Section 3.6.2) This usage of the word "canonical" is related to the mathematical concept of "canonical form".

CNAME: "It is traditional to refer to the owner of a CNAME record as 'a CNAME'. This is unfortunate, as 'CNAME' is an abbreviation of 'canonical name', and the owner of a CNAME record is an alias, not a canonical name." (Quoted from [RFC2181], Section 10.1.1)

Public suffix: "A domain that is controlled by a public registry." (Quoted from [RFC6265], Section 5.3) A common definition for this term is a domain under which subdomains can be registered, and on which HTTP cookies ([RFC6265]) should not be set. There is no indication in a domain name whether it is a public suffix; that can only be determined by outside means. In fact, both a domain and a subdomain of that domain can be public suffixes.

There is nothing inherent in a domain name to indicate whether it is a public suffix. One resource for identifying public suffixes is the Public Suffix List (PSL) maintained by Mozilla (http://publicsuffix.org/).

For example, at the time this document is published, the "com.au" domain is listed as a public suffix in the PSL. (Note that this example might change in the future.)

Note that the term "public suffix" is controversial in the DNS community for many reasons, and may be significantly changed in the future. One example of the difficulty of calling a domain a public suffix is that designation can change over time as the registration policy for the zone changes, such as was the case with the "uk" TLD in 2014.

3. DNS Header and Response Codes

The header of a DNS message is its first 12 octets. Many of the fields and flags in the header diagram in Sections 4.1.1 through 4.1.3 of [RFC1035] are referred to by their names in that diagram. For example, the response codes are called "RCODEs", the data for a record is called the "RDATA", and the authoritative answer bit is often called "the AA flag" or "the AA bit".

QNAME The most commonly-used rough definition is that the QNAME is a field in the Question section of a query. "A standard query specifies a target domain name (QNAME), query type (QTYPE), and query class (QCLASS) and asks for RRs which match." (Quoted from [RFC1034], Section 3.7.1.). Strictly speaking, the definition comes from [RFC1035], Section 4.1.2, where the QNAME is defined in respect of the Question Section. This definition appears to be applied consistently: the discussion of inverse queries in section 6.4 refers to the "owner name of the query RR and its TTL", because inverse queries populate the Answer Section and leave the Question Section empty. (Inverse queries are deprecated in [RFC3425], and so relevant definitions do not appear in this document.)

[RFC2308], however, has an alternate definition that puts the QNAME in the answer (or series of answers) instead of the query. It defines QNAME as: "...the name in the query section of an answer, or where this resolves to a CNAME, or CNAME chain, the data field of the last CNAME. The last CNAME in this sense is that which contains a value which does not resolve to another CNAME." This definition has a certain internal logic, because of the way CNAME substitution works and the definition of CNAME. If a name server does not find an RRset that matches a query, but it finds the same name in the same class with a CNAME record, then the name server "includes the CNAME record in the response and restarts the query at the domain name specified in the data field of the CNAME record." ([RFC1034] Section 3.6.2). This is made explicit in the resolution algorithm outlined in Section 4.3.2 of [RFC1034], which says to "change QNAME to the canonical name in the CNAME RR, and go back to step 1" in the case of a CNAME RR. Since a CNAME record explicitly declares that the owner name is canonically named what is in the RDATA, then there is a way to

view the new name (i.e. the name that was in the RDATA of the CNAME RR) as also being the QNAME.

This creates a kind of confusion, however, because the answer to a query that results in CNAME processing contains in the echoed Question Section one QNAME (the name in the original query), and a second QNAME that is in the data field of the last CNAME. The confusion comes from the iterative/recursive mode of resolution, which finally returns an answer that need not actually have the same owner name as the QNAME contained in the original query.

To address this potential confusion, it is helpful to distinguish between three meanings:

* QNAME (original): The name actually sent in the Question Section in the orignal query, which is always echoed in the (final) reply in the Question Section when the QR bit is set to 1.

* QNAME (effective): A name actually resolved, which is either the name originally queried, or a name received in a CNAME chain response.

* QNAME (final): The name actually resolved, which is either the name actually queried or else the last name in a CNAME chain response.

Some of response codes that are defined in [RFC1035] have acquired their own shorthand names. Some common response code names that appear without reference to the numeric value are "FORMERR", "SERVFAIL", and "NXDOMAIN" (the latter of which is also referred to as "Name Error"). All of the RCODEs are listed at http://www.iana.org/assignments/dns-parameters, although that site uses mixed-case capitalization, while most documents use all-caps.

NODATA: "A pseudo RCODE which indicates that the name is valid for the given class, but there are no records of the given type. A NODATA response has to be inferred from the answer." (Quoted from [RFC2308], Section 1.) "NODATA is indicated by an answer with the RCODE set to NOERROR and no relevant answers in the answer section. The authority section will contain an SOA record, or there will be no NS records there." (Quoted from [RFC2308], Section 2.2.) Note that referrals have a similar format to NODATA replies; [RFC2308] explains how to distinguish them.

The term "NXRRSET" is sometimes used as a synonym for NODATA. However, this is a mistake, given that NXRRSET is a specific error code defined in [RFC2136].

Negative response: A response that indicates that a particular RRset does not exist, or whose RCODE indicates the nameserver cannot answer. Sections 2 and 7 of [RFC2308] describe the types of negative responses in detail.

Referrals: Data from the authority section of a non-authoritative answer. [RFC1035] Section 2.1 defines "authoritative" data. However, referrals at zone cuts (defined in Section 6) are not authoritative. Referrals may be zone cut NS resource records and their glue records. NS records on the parent side of a zone cut are an authoritative delegation, but are normally not treated as authoritative data. In general, a referral is a way for a server to send an answer saying that the server does not know the answer, but knows where the query should be directed in order to get an answer. Historically, many authoritative servers answered with a referral to the root zone when queried for a name for which they were not authoritative, but this practice has declined.

4. Resource Records

RR: An acronym for resource record. ([RFC1034], Section 3.6.)

RRset: A set of resource records with the same label, class and type, but with different data. (Definition from [RFC2181]) Also spelled RRSet in some documents. As a clarification, "same label" in this definition means "same owner name". In addition, [RFC2181] states that "the TTLs of all RRs in an RRSet must be the same". (This definition is definitely not the same as "the response one gets to a query for QTYPE=ANY", which is an unfortunate misunderstanding.)

Master file: "Master files are text files that contain RRs in text form. Since the contents of a zone can be expressed in the form of a list of RRs a master file is most often used to define a zone, though it can be used to list a cache’s contents." ([RFC1035], Section 5.)

Presentation format: The text format used in master files. This format is shown but not formally defined in [RFC1034] and [RFC1035]. The term "presentation format" first appears in [RFC4034].

EDNS: The extension mechanisms for DNS, defined in [RFC6891]. Sometimes called "EDNS0" or "EDNS(0)" to indicate the version number. EDNS allows DNS clients and servers to specify message sizes larger than the original 512 octet limit, to expand the response code space, and potentially to carry additional options that affect the handling of a DNS query.

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

Owner: The domain name where a RR is found ([RFC1034], Section 3.6). Often appears in the term "owner name".

SOA field names: DNS documents, including the definitions here, often refer to the fields in the RDATA of an SOA resource record by field name. Those fields are defined in Section 3.3.13 of [RFC1035]. The names (in the order they appear in the SOA RDATA) are MNAME, RNAME, SERIAL, REFRESH, RETRY, EXPIRE, and MINIMUM. Note that the meaning of MINIMUM field is updated in Section 4 of [RFC2308]; the new definition is that the MINIMUM field is only "the TTL to be used for negative responses". This document tends to use field names instead of terms that describe the fields.

TTL: The maximum "time to live" of a resource record. "A TTL value is an unsigned number, with a minimum value of 0, and a maximum value of 2147483647. That is, a maximum of 2^31 - 1. When transmitted, the TTL is encoded in the less significant 31 bits of the 32 bit TTL field, with the most significant, or sign, bit set to zero." (Quoted from [RFC2181], Section 8) (Note that [RFC1035] erroneously stated that this is a signed integer; that was fixed by [RFC2181].)

The TTL "specifies the time interval that the resource record may be cached before the source of the information should again be consulted". (Quoted from [RFC1035], Section 3.2.1) Also: "the time interval (in seconds) that the resource record may be cached before it should be discarded". (Quoted from [RFC1035], Section 4.1.3). Despite being defined for a resource record, the TTL of every resource record in an RRset is required to be the same ([RFC2181], Section 5.2).

The reason that the TTL is the maximum time to live is that a cache operator might decide to shorten the time to live for operational purposes, such as if there is a policy to disallow TTL values over a certain number. Also, if a value is flushed from the cache when its value is still positive, the value effectively becomes zero. Some servers are known to ignore the TTL on some RRsets (such as when the authoritative data has a very short TTL) even though this is against the advice in RFC 1035.

There is also the concept of a "default TTL" for a zone, which can be a configuration parameter in the server software. This is often expressed by a default for the entire server, and a default

for a zone using the $TTL directive in a zone file. The $TTL directive was added to the master file format by [RFC2308].

Class independent: A resource record type whose syntax and semantics are the same for every DNS class. A resource record type that is not class independent has different meanings depending on the DNS class of the record, or the meaning is undefined for classes other than IN (class 1, the Internet).

5. DNS Servers and Clients

This section defines the terms used for the systems that act as DNS clients, DNS servers, or both. In the RFCs, DNS servers are sometimes called "name servers", "nameservers", or just "servers". There is no formal definition of DNS server, but the RFCs generally assume that it is an Internet server that listens for queries and sends responses using the DNS protocol defined in [RFC1035] and its successors.

For terminology specific to the public DNS root server system, see [RSSAC026]. That document defines terms such as "root server", "root server operator", and terms that are specific to the way that the root zone of the public DNS is served.

Resolver: A program "that extract[s] information from name servers in response to client requests." (Quoted from [RFC1034], Section 2.4) "The resolver is located on the same machine as the program that requests the resolver's services, but it may need to consult name servers on other hosts." (Quoted from [RFC1034], Section 5.1) A resolver performs queries for a name, type, and class, and receives answers. The logical function is called "resolution". In practice, the term is usually referring to some specific type of resolver (some of which are defined below), and understanding the use of the term depends on understanding the context.

A related term is "resolve", which is not formally defined in [RFC1034] or [RFC1035]. An imputed definition might be "asking a question that consists of a domain name, class, and type, and receiving some sort of answer". Similarly, an imputed definition of "resolution" might be "the answer received from resolving".

Stub resolver: A resolver that cannot perform all resolution itself. Stub resolvers generally depend on a recursive resolver to undertake the actual resolution function. Stub resolvers are discussed but never fully defined in Section 5.3.1 of [RFC1034]. They are fully defined in Section 6.1.3.1 of [RFC1123].

Iterative mode: A resolution mode of a server that receives DNS queries and responds with a referral to another server. Section 2.3 of [RFC1034] describes this as "The server refers the client to another server and lets the client pursue the query". A resolver that works in iterative mode is sometimes called an "iterative resolver".

Recursive mode: A resolution mode of a server that receives DNS queries and either responds to those queries from a local cache or sends queries to other servers in order to get the final answers to the original queries. Section 2.3 of [RFC1034] describes this as "The first server pursues the query for the client at another server". A server operating in recursive mode may be thought of as having a name server side (which is what answers the query) and a resolver side (which performs the resolution function). Systems operating in this mode are commonly called "recursive servers". Sometimes they are called "recursive resolvers". While strictly the difference between these is that one of them sends queries to another recursive server and the other does not, in practice it is not possible to know in advance whether the server that one is querying will also perform recursion; both terms can be observed in use interchangeably.

Full resolver: This term is used in [RFC1035], but it is not defined there. RFC 1123 defines a "full-service resolver" that may or may not be what was intended by "full resolver" in [RFC1035]. This term is not properly defined in any RFC.

Full-service resolver: Section 6.1.3.1 of [RFC1123] defines this term to mean a resolver that acts in recursive mode with a cache (and meets other requirements).

Recursive resolver: A resolver that acts in recursive mode. In general, a recursive resolver is expected to cache the answers it receives (which would make it a full-service resolver), but some recursive resolvers might not cache.

Priming: "The act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers." (Quoted from [RFC8109], Section 2.) Priming is most often done from a configuration setting that contains a list of authoritative servers for the root zone.

Root hints: "Operators who manage a DNS recursive resolver typically need to configure a 'root hints file'. This file contains the names and IP addresses of the authoritative name servers for the root zone, so the software can bootstrap the DNS resolution

process. For many pieces of software, this list comes built into the software." (Quoted from [IANA_RootFiles])

Negative caching: "The storage of knowledge that something does not exist, cannot give an answer, or does not give an answer." (Quoted from [RFC2308], Section 1)

Authoritative server: "A server that knows the content of a DNS zone from local knowledge, and thus can answer queries about that zone without needing to query other servers." (Quoted from [RFC2182], Section 2.) It is a system that responds to DNS queries with information about zones for which it has been configured to answer with the AA flag in the response header set to 1. It is a server that has authority over one or more DNS zones. Note that it is possible for an authoritative server to respond to a query without the parent zone delegating authority to that server. Authoritative servers also provide "referrals", usually to child zones delegated from them; these referrals have the AA bit set to 0 and come with referral data in the Authority and (if needed) the Additional sections.

Authoritative-only server: A name server that only serves authoritative data and ignores requests for recursion. It will "not normally generate any queries of its own. Instead, it answers non-recursive queries from iterative resolvers looking for information in zones it serves." (Quoted from [RFC4697], Section 2.4)

Zone transfer: The act of a client requesting a copy of a zone and an authoritative server sending the needed information. (See Section 6 for a description of zones.) There are two common standard ways to do zone transfers: the AXFR ("Authoritative Transfer") mechanism to copy the full zone (described in [RFC5936], and the IXFR ("Incremental Transfer") mechanism to copy only parts of the zone that have changed (described in [RFC1995]). Many systems use non-standard methods for zone transfer outside the DNS protocol.

Secondary server: "An authoritative server which uses zone transfer to retrieve the zone" (Quoted from [RFC1996], Section 2.1). [RFC2182] describes secondary servers in detail. Although early DNS RFCs such as [RFC1996] referred to this as a "slave", the current common usage has shifted to calling it a "secondary". Secondary servers are also discussed in [RFC1034].

Slave server: See secondary server.

Primary server: "Any authoritative server configured to be the source of zone transfer for one or more [secondary] servers" (Quoted from [RFC1996], Section 2.1) or, more specifically, "an authoritative server configured to be the source of AXFR or IXFR data for one or more [secondary] servers" (Quoted from [RFC2136]). Although early DNS RFCs such as [RFC1996] referred to this as a "master", the current common usage has shifted to "primary". Primary servers are also discussed in [RFC1034].

Master server: See primary server.

Primary master: "The primary master is named in the zone's SOA MNAME field and optionally by an NS RR". (Quoted from [RFC1996], Section 2.1). [RFC2136] defines "primary master" as "Master server at the root of the AXFR/IXFR dependency graph. The primary master is named in the zone's SOA MNAME field and optionally by an NS RR. There is by definition only one primary master server per zone." The idea of a primary master is only used by [RFC2136], and is considered archaic in other parts of the DNS.

Stealth server: This is "like a slave server except not listed in an NS RR for the zone." (Quoted from [RFC1996], Section 2.1)

Hidden master: A stealth server that is a primary server for zone transfers. "In this arrangement, the master name server that processes the updates is unavailable to general hosts on the Internet; it is not listed in the NS RRset." (Quoted from [RFC6781], Section 3.4.3). An earlier RFC, [RFC4641], said that the hidden master's name "appears in the SOA RRs MNAME field", although in some setups, the name does not appear at all in the public DNS. A hidden master can also be a secondary server itself.

Forwarding: The process of one server sending a DNS query with the RD bit set to 1 to another server to resolve that query. Forwarding is a function of a DNS resolver; it is different than simply blindly relaying queries.

[RFC5625] does not give a specific definition for forwarding, but describes in detail what features a system that forwards need to support. Systems that forward are sometimes called "DNS proxies", but that term has not yet been defined (even in [RFC5625]).

Forwarder: Section 1 of [RFC2308] describes a forwarder as "a nameserver used to resolve queries instead of directly using the authoritative nameserver chain". [RFC2308] further says "The forwarder typically either has better access to the internet, or maintains a bigger cache which may be shared amongst many

resolvers." That definition appears to suggest that forwarders normally only query authoritative servers. In current use, however, forwarders often stand between stub resolvers and recursive servers. [RFC2308] is silent on whether a forwarder is iterative-only or can be a full-service resolver.

Policy-implementing resolver: A resolver acting in recursive mode that changes some of the answers that it returns based on policy criteria, such as to prevent access to malware sites or objectionable content. In general, a stub resolver has no idea whether upstream resolvers implement such policy or, if they do, the exact policy about what changes will be made. In some cases, the user of the stub resolver has selected the policy-implementing resolver with the explicit intention of using it to implement the policies. In other cases, policies are imposed without the user of the stub resolver being informed.

Open resolver: A full-service resolver that accepts and processes queries from any (or nearly any) stub resolver. This is sometimes also called a "public resolver", although the term "public resolver" is used more with open resolvers that are meant to be open, as compared to the vast majority of open resolvers that are probably misconfigured to be open. Open resolvers are discussed in [RFC5358]

View: A configuration for a DNS server that allows it to provide different answers depending on attributes of the query. Typically, views differ by the source IP address of a query, but can also be based on the destination IP address, the type of query (such as AXFR), whether it is recursive, and so on. Views are often used to provide more names or different addresses to queries from "inside" a protected network than to those "outside" that network. Views are not a standardized part of the DNS, but they are widely implemented in server software.

Passive DNS: A mechanism to collect DNS data by storing DNS transactions from name servers. Some of these systems also collect the DNS queries associated with the responses. Passive DNS databases can be used to answer historical questions about DNS zones such as which answers were witnessed at what times in the past. Passive DNS databases allow searching of the stored records on keys other than just the name and type, such as "find all names which have A records of a particular value".

Anycast: "The practice of making a particular service address available in multiple, discrete, autonomous locations, such that datagrams sent are routed to one of several available locations." (Quoted from [RFC4786], Section 2)

Instance: "When anycast routing is used to allow more than one server to have the same IP address, each one of those servers is commonly referred to as an 'instance'." "An instance of a server, such as a root server, is often referred to as an 'Anycast instance'." (Quoted from [RSSAC026])

Split DNS: "Where a corporate network serves up partly or completely different DNS inside and outside its firewall. There are many possible variants on this; the basic point is that the correspondence between a given FQDN (fully qualified domain name) and a given IPv4 address is no longer universal and stable over long periods." (Quoted from [RFC2775], Section 3.8)

6. Zones

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

Zone: "Authoritative information is organized into units called 'zones', and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone." (Quoted from [RFC1034], Section 2.4)

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

Parent: "The domain in which the Child is registered." (Quoted from [RFC7344], Section 1.1) Earlier, "parent name server" was defined in [RFC0882] as "the name server that has authority over the place in the domain name space that will hold the new domain". (Note that [RFC0882] was obsoleted by [RFC1034] and [RFC1035].) [RFC0819] also has some description of the relationship between parents and children.

Origin:

(a) "The domain name that appears at the top of a zone (just below the cut that separates the zone from its parent). The name of the zone is the same as the name of the domain at the zone's origin." (Quoted from [RFC2181], Section 6.) These days, this sense of "origin" and "apex" (defined below) are often used interchangeably.

(b) The domain name within which a given relative domain name appears in zone files. Generally seen in the context of "$ORIGIN", which is a control entry defined in [RFC1035], Section 5.1, as part of the master file format. For example, if

the $ORIGIN is set to "example.org.", then a master file line for "www" is in fact an entry for "www.example.org.".

Apex: The point in the tree at an owner of an SOA and corresponding authoritative NS RRset. This is also called the "zone apex". [RFC4033] defines it as "the name at the child's side of a zone cut". The "apex" can usefully be thought of as a data-theoretic description of a tree structure, and "origin" is the name of the same concept when it is implemented in zone files. The distinction is not always maintained in use, however, and one can find uses that conflict subtly with this definition. [RFC1034] uses the term "top node of the zone" as a synonym of "apex", but that term is not widely used. These days, the first sense of "origin" (above) and "apex" are often used interchangeably.

Zone cut: The delimitation point between two zones where the origin of one of the zones is the child of the other zone.

"Zones are delimited by 'zone cuts'. Each zone cut separates a 'child' zone (below the cut) from a 'parent' zone (above the cut). (Quoted from [RFC2181], Section 6; note that this is barely an ostensive definition.) Section 4.2 of [RFC1034] uses "cuts" as 'zone cut'."

Delegation: The process by which a separate zone is created in the name space beneath the apex of a given domain. Delegation happens when an NS RRset is added in the parent zone for the child origin. Delegation inherently happens at a zone cut. The term is also commonly a noun: the new zone that is created by the act of delegating.

Glue records: "[Resource records] which are not part of the authoritative data [of the zone], and are address resource records for the [name servers in subzones]. These RRs are only necessary if the name server's name is 'below' the cut, and are only used as part of a referral response." Without glue "we could be faced with the situation where the NS RRs tell us that in order to learn a name server's address, we should contact the server using the address we wish to learn." (Definition from [RFC1034], Section 4.2.1)

A later definition is that glue "includes any record in a zone file that is not properly part of that zone, including nameserver records of delegated sub-zones (NS records), address records that accompany those NS records (A, AAAA, etc), and any other stray data that might appear" ([RFC2181], Section 5.4.1). Although glue is sometimes used today with this wider definition in mind, the context surrounding the [RFC2181] definition suggests it is

intended to apply to the use of glue within the document itself and not necessarily beyond.

In-bailiwick: An adjective to describe a name server whose name is either subordinate to or (rarely) the same as the zone origin. In-bailiwick name servers may have glue records in their parent zone (using the first of the definitions of "glue records" in the definition above). "In-bailiwick" names are divided into two type of name server names: "in-domain" names and "sibling domain" names:

* In-domain -- an adjective to describe a name server whose name is either subordinate to or (rarely) the same as the owner name of the NS resource records. An in-domain name server name MUST have glue records or name resolution fails. For example, a delegation for "child.example.com" may have "in-domain" name server name "ns.child.example.com".

* Sibling domain: -- a name server’s name that is either subordinate to or (rarely) the same as the zone origin and not subordinate to or the same as the owner name of the NS resource records. Glue records for sibling domains are allowed, but not necessary. For example, a delegation for "child.example.com" in "example.com" zone may have "sibling" name server name "ns.another.example.com".

Out-of-bailiwick: The antonym of in-bailiwick. An adjective to describe a name server whose name is not subordinate to or the same as the zone origin. Glue records for out-of-bailiwick name servers are useless.

Authoritative data: "All of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone." (Quoted from [RFC1034], Section 4.2.1) It is noted that this definition might inadvertently also include any NS records that appear in the zone, even those that might not truly be authoritative because there are identical NS RRs below the zone cut. This reveals the ambiguity in the notion of authoritative data, because the parent-side NS records authoritatively indicate the delegation, even though they are not themselves authoritative data.

Root zone: The zone of a DNS-based tree whose apex is the zero-length label. Also sometimes called "the DNS root".

Empty non-terminals (ENT): "Domain names that own no resource records but have subdomains that do." (Quoted from [RFC4592], Section 2.2.2.) A typical example is in SRV records: in the name

"_sip._tcp.example.com", it is likely that "_tcp.example.com" has no RRsets, but that "_sip._tcp.example.com" has (at least) an SRV RRset.

Delegation-centric zone: A zone that consists mostly of delegations to child zones. This term is used in contrast to a zone that might have some delegations to child zones, but also has many data resource records for the zone itself and/or for child zones. The term is used in [RFC4956] and [RFC5155], but is not defined there.

Wildcard: [RFC1034] defined "wildcard", but in a way that turned out to be confusing to implementers. Special treatment is given to RRs with owner names starting with the label "*". "Such RRs are called 'wildcards'. Wildcard RRs can be thought of as instructions for synthesizing RRs." (Quoted from [RFC1034], Section 4.3.3) For an extended discussion of wildcards, including clearer definitions, see [RFC4592].

Asterisk label: "The first octet is the normal label type and length for a 1-octet-long label, and the second octet is the ASCII representation for the '*' character. A descriptive name of a label equaling that value is an 'asterisk label'." (Quoted from [RFC4592], Section 2.1.1)

Wildcard domain name: "A 'wildcard domain name' is defined by having its initial (i.e., leftmost or least significant) label be asterisk label." (Quoted from [RFC4592], Section 2.1.1)

Closest encloser: "The longest existing ancestor of a name." (Quoted from [RFC5155], Section 1.3) An earlier definition is "The node in the zone's tree of existing domain names that has the most labels matching the query name (consecutively, counting from the root label downward). Each match is a 'label match' and the order of the labels is the same." (Quoted from [RFC4592], Section 3.3.1)

Closest provable encloser: "The longest ancestor of a name that can be proven to exist. Note that this is only different from the closest encloser in an Opt-Out zone." (Quoted from [RFC5155], Section 1.3)

Next closer name: "The name one label longer than the closest provable encloser of a name." (Quoted from [RFC5155], Section 1.3)

Source of Synthesis: "The source of synthesis is defined in the context of a query process as that wildcard domain name immediately descending from the closest encloser, provided that

this wildcard domain name exists. 'Immediately descending' means that the source of synthesis has a name of the form: <asterisk label>.<closest encloser>." (Quoted from [RFC4592], Section 3.3.1)

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

Fast flux DNS: This "occurs when a domain is found in DNS using A records to multiple IP addresses, each of which has a very short Time-to-Live (TTL) value associated with it. This means that the domain resolves to varying IP addresses over a short period of time." (Quoted from [RFC6561], Section 1.1.5, with typo corrected) It is often used to deliver malware. Because the addresses change so rapidly, it is difficult to ascertain all the hosts. It should be noted that the technique also works with AAAA records, but such use is not frequently observed on the Internet as of this writing.

Reverse DNS, reverse lookup: "The process of mapping an address to a name is generally known as a 'reverse lookup', and the IN-ADDR.ARPA and IP6.ARPA zones are said to support the 'reverse DNS'." (Quoted from [RFC5855], Section 1)

Forward lookup: "Hostname-to-address translation". (Quoted from [RFC2133], Section 6)

arpa: Address and Routing Parameter Area Domain: "The 'arpa' domain was originally established as part of the initial deployment of the DNS, to provide a transition mechanism from the Host Tables that were common in the ARPANET, as well as a home for the IPv4 reverse mapping domain. During 2000, the abbreviation was redesignated to 'Address and Routing Parameter Area' in the hope of reducing confusion with the earlier network name." (Quoted from [RFC3172], Section 2.)

Infrastructure domain: A domain whose "role is to support the operating infrastructure of the Internet". (Quoted from [RFC3172], Section 2.)

Service name: "Service names are the unique key in the Service Name and Transport Protocol Port Number registry. This unique symbolic name for a service may also be used for other purposes, such as in DNS SRV records." (Quoted from [RFC6335], Section 5.)

7. Registration Model

Registry: The administrative operation of a zone that allows registration of names within that zone. People often use this term to refer only to those organizations that perform registration in large delegation-centric zones (such as TLDs); but formally, whoever decides what data goes into a zone is the registry for that zone. This definition of "registry" is from a DNS point of view; for some zones, the policies that determine what can go in the zone are decided by superior zones and not the registry operator.

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

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

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

WHOIS: A protocol specified in [RFC3912], often used for querying registry databases. WHOIS data is frequently used to associate registration data (such as zone management contacts) with domain names. The term "WHOIS data" is often used as a synonym for the registry database, even though that database may be served by different protocols, particularly RDAP. The WHOIS protocol is also used with IP address registry data.

RDAP: The Registration Data Access Protocol, defined in [RFC7480], [RFC7481], [RFC7482], [RFC7483], [RFC7484], and [RFC7485]. The RDAP protocol and data format are meant as a replacement for WHOIS.

DNS operator: An entity responsible for running DNS servers. For a zone’s authoritative servers, the registrant may act as their own DNS operator, or their registrar may do it on their behalf, or they may use a third-party operator. For some zones, the registry function is performed by the DNS operator plus other entities who decide about the allowed contents of the zone.

8. General DNSSEC

Most DNSSEC terms are defined in [RFC4033], [RFC4034], [RFC4035], and [RFC5155]. The terms that have caused confusion in the DNS community are highlighted here.

DNSSEC-aware and DNSSEC-unaware: These two terms, which are used in some RFCs, have not been formally defined. However, Section 2 of [RFC4033] defines many types of resolvers and validators, including "non-validating security-aware stub resolver", "non-validating stub resolver", "security-aware name server", "security-aware recursive name server", "security-aware resolver", "security-aware stub resolver", and "security-oblivious 'anything'". (Note that the term "validating resolver", which is used in some places in DNSSEC-related documents, is also not defined in those RFCs, but is defined below.)

Signed zone: "A zone whose RRsets are signed and that contains properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records." (Quoted from [RFC4033], Section 2.) It has been noted in other contexts that the zone itself is not really signed, but all the relevant RRsets in the zone are signed. Nevertheless, if a zone that should be signed contains any RRsets that are not signed (or opted out), those RRsets will be treated as bogus, so the whole zone needs to be handled in some way.

It should also be noted that, since the publication of [RFC6840], NSEC records are no longer required for signed zones: a signed zone might include NSEC3 records instead. [RFC7129] provides additional background commentary and some context for the NSEC and NSEC3 mechanisms used by DNSSEC to provide authenticated denial-of-existence responses. NSEC and NSEC3 are described below.

Unsigned zone: Section 2 of [RFC4033] defines this as "a zone that is not signed". Section 2 of [RFC4035] defines this as "A zone that does not include these records [properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records] according to the rules in this section". There is an important note at the end of Section 5.2 of [RFC4035] that defines an additional situation in which a zone is considered unsigned: "If the resolver does not support any of the algorithms listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned."

NSEC: "The NSEC record allows a security-aware resolver to authenticate a negative reply for either name or type non-existence with the same mechanisms used to authenticate other DNS replies." (Quoted from [RFC4033], Section 3.2.) In short, an NSEC record provides authenticated denial of existence.

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

NSEC3: Like the NSEC record, the NSEC3 record also provides authenticated denial of existence; however, NSEC3 records mitigate against zone enumeration and support Opt-Out. NSEC resource records require associated NSEC3PARAM resource records. NSEC3 and NSEC3PARAM resource records are defined in [RFC5155].

Note that [RFC6840] says that [RFC5155] "is now considered part of the DNS Security Document Family as described by Section 10 of [RFC4033]". This means that some of the definitions from earlier RFCs that only talk about NSEC records should probably be considered to be talking about both NSEC and NSEC3.

Opt-out: "The Opt-Out Flag indicates whether this NSEC3 RR may cover unsigned delegations." (Quoted from [RFC5155], Section 3.1.2.1.) Opt-out tackles the high costs of securing a delegation to an insecure zone. When using Opt-Out, names that are an insecure delegation (and empty non-terminals that are only derived from insecure delegations) don't require an NSEC3 record or its corresponding RRSIG records. Opt-Out NSEC3 records are not able to prove or deny the existence of the insecure delegations. (Adapted from [RFC7129], Section 5.1)

Insecure delegation: "A signed name containing a delegation (NS RRset), but lacking a DS RRset, signifying a delegation to an unsigned subzone." (Quoted from [RFC4956], Section 2.)

Zone enumeration: "The practice of discovering the full content of a zone via successive queries." (Quoted from [RFC5155], Section 1.3.) This is also sometimes called "zone walking". Zone enumeration is different from zone content guessing where the guesser uses a large dictionary of possible labels and sends successive queries for them, or matches the contents of NSEC3 records against such a dictionary.

Validation: Validation, in the context of DNSSEC, refers to the following:

* Checking the validity of DNSSEC signatures

* Checking the validity of DNS responses, such as those including authenticated denial of existence

* Building an authentication chain from a trust anchor to a DNS response or individual DNS RRsets in a response

The first two definitions above consider only the validity of individual DNSSEC components such as the RRSIG validity or NSEC proof validity. The third definition considers the components of the entire DNSSEC authentication chain, and thus requires "configured knowledge of at least one authenticated DNSKEY or DS RR" (as described in [RFC4035], Section 5).

[RFC4033], Section 2, says that a "Validating Security-Aware Stub Resolver... performs signature validation" and uses a trust anchor "as a starting point for building the authentication chain to a signed DNS response", and thus uses the first and third definitions above. The process of validating an RRSIG resource record is described in [RFC4035], Section 5.3.

[RFC5155] refers to validating responses throughout the document, in the context of hashed authenticated denial of existence; this uses the second definition above.

The term "authentication" is used interchangeably with "validation", in the sense of the third definition above. [RFC4033], Section 2, describes the chain linking trust anchor to DNS data as the "authentication chain". A response is considered to be authentic if "all RRsets in the Answer and Authority sections of the response [are considered] to be authentic" ([RFC4035]). DNS data or responses deemed to be authentic or validated have a security status of "secure" ([RFC4035], Section 4.3; [RFC4033], Section 5). "Authenticating both DNS keys and data is a matter of local policy, which may extend or even override the [DNSSEC] protocol extensions" ([RFC4033], Section 3.1).

The term "verification", when used, is usually synonym for "validation".

Validating resolver: A security-aware recursive name server, security-aware resolver, or security-aware stub resolver that is applying at least one of the definitions of validation (above), as appropriate to the resolution context. For the same reason that the generic term "resolver" is sometimes ambiguous and needs to be

evaluated in context (see Section 5), "validating resolver" is a context-sensitive term.

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

Zone signing key (ZSK): "DNSSEC keys that can be used to sign all the RRsets in a zone that require signatures, other than the apex DNSKEY RRset." (Quoted from [RFC6781], Section 3.1) Note that the roles KSK and ZSK are not mutually exclusive: a single key can be both KSK and ZSK at the same time. Also note that a ZSK is sometimes used to sign the apex DNSKEY RRset.

Combined signing key (CSK): "In cases where the differentiation between the KSK and ZSK is not made, i.e., where keys have the role of both KSK and ZSK, we talk about a Single-Type Signing Scheme." (Quoted from [RFC6781], Section 3.1) This is sometimes called a "combined signing key" or CSK. It is operational practice, not protocol, that determines whether a particular key is a ZSK, a KSK, or a CSK.

Secure Entry Point (SEP): A flag in the DNSKEY RDATA that "can be used to distinguish between keys that are intended to be used as the secure entry point into the zone when building chains of trust, i.e., they are (to be) pointed to by parental DS RRs or configured as a trust anchor. Therefore, it is suggested that the SEP flag be set on keys that are used as KSKs and not on keys that are used as ZSKs, while in those cases where a distinction between a KSK and ZSK is not made (i.e., for a Single-Type Signing Scheme), it is suggested that the SEP flag be set on all keys." (Quoted from [RFC6781], Section 3.2.3.) Note that the SEP flag is only a hint, and its presence or absence may not be used to disqualify a given DNSKEY RR from use as a KSK or ZSK during validation.

The original defintion of SEPs was in [RFC3757]. That definition clearly indicated that the SEP was a key, not just a bit in the key. The abstract of [RFC3757] says: "With the Delegation Signer (DS) resource record (RR), the concept of a public key acting as a secure entry point (SEP) has been introduced. During exchanges of public keys with the parent there is a need to differentiate SEP keys from other public keys in the Domain Name System KEY (DNSKEY) resource record set. A flag bit in the DNSKEY RR is defined to indicate that DNSKEY is to be used as a SEP." That definition of the SEP as a key was made obsolete by [RFC4034], and the definition from [RFC6781] is consistent with [RFC4034].

Trust anchor: "A configured DNSKEY RR or DS RR hash of a DNSKEY RR. A validating security-aware resolver uses this public key or hash as a starting point for building the authentication chain to a signed DNS response." (Quoted from [RFC4033], Section 2)

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

DNSSEC Practice Statement (DPS): "A practices disclosure document that may support and be a supplemental document to the DNSSEC Policy (if such exists), and it states how the management of a given zone implements procedures and controls at a high level." (Quoted from [RFC6841], Section 2)

Hardware security module (HSM): A specialized piece of hardware that is used to create keys for signatures and to sign messages. In DNSSEC, HSMs are often used to hold the private keys for KSKs and ZSKs and to create the RRSIG records at periodic intervals.

Signing software: Authoritative DNS servers that supports DNSSEC often contains software that facilitates the creation and maintenance of DNSSEC signatures in zones. There is also stand-alone software that can be used to sign a zone regardless of whether the authoritative server itself supports signing. Sometimes signing software can support particular HSMs as part of the signing process.

9. DNSSEC States

A validating resolver can determine that a response is in one of four states: secure, insecure, bogus, or indeterminate. These states are defined in [RFC4033] and [RFC4035], although the two definitions differ a bit. This document makes no effort to reconcile the two definitions, and takes no position as to whether they need to be reconciled.

Section 5 of [RFC4033] says:

A validating resolver can determine the following 4 states:

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

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

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

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

Section 4.3 of [RFC4035] says:

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

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

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

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

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

10. Security Considerations

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

11. IANA Considerations

None.

12. References

12.1. Normative References

[IANA_RootFiles]
Internet Assigned Numbers Authority, "IANA Root Files", 2016, <http://www.iana.org/domains/root/files>.

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

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

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[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>.

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

[RFC2182] Elz, R., Bush, R., Bradner, S., and M. Patton, "Selection and Operation of Secondary DNS Servers", BCP 16, RFC 2182, DOI 10.17487/RFC2182, July 1997, <https://www.rfc-editor.org/info/rfc2182>.

[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>.

[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>.

[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>.

[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>.

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

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5855] Abley, J. and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", BCP 155, RFC 5855, DOI 10.17487/RFC5855, May 2010, <https://www.rfc-editor.org/info/rfc5855>.

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

[RFC6561] Livingood, J., Mody, N., and M. O'Reirdan, "Recommendations for the Remediation of Bots in ISP Networks", RFC 6561, DOI 10.17487/RFC6561, March 2012, <https://www.rfc-editor.org/info/rfc6561>.

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

[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>.

[RFC6841] Ljunggren, F., Eklund Lowinder, AM., and T. Okubo, "A Framework for DNSSEC Policies and DNSSEC Practice Statements", RFC 6841, DOI 10.17487/RFC6841, January 2013, <https://www.rfc-editor.org/info/rfc6841>.

[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>.

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

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

12.2. Informative References

[IANA_Resource_Registry] Internet Assigned Numbers Authority, "Resource Record (RR) TYPEs", 2017, <http://www.iana.org/assignments/dns-parameters/>.

[RFC0819] Su, Z. and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC 819, DOI 10.17487/RFC0819, August 1982, <https://www.rfc-editor.org/info/rfc819>.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

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

[RFC2133] Gilligan, R., Thomson, S., Bound, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 2133, DOI 10.17487/RFC2133, April 1997, <https://www.rfc-editor.org/info/rfc2133>.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <https://www.rfc-editor.org/info/rfc2775>.

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[RFC3425] Lawrence, D., "Obsoleting IQUERY", RFC 3425, DOI 10.17487/RFC3425, November 2002, <https://www.rfc-editor.org/info/rfc3425>.

[RFC3757] Kolkman, O., Schlyter, J., and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag", RFC 3757, DOI 10.17487/RFC3757, April 2004, <https://www.rfc-editor.org/info/rfc3757>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, DOI 10.17487/RFC4641, September 2006, <https://www.rfc-editor.org/info/rfc4641>.

[RFC4697] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", BCP 123, RFC 4697, DOI 10.17487/RFC4697, October 2006, <https://www.rfc-editor.org/info/rfc4697>.

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

[RFC4956] Arends, R., Kosters, M., and D. Blacka, "DNS Security (DNSSEC) Opt-In", RFC 4956, DOI 10.17487/RFC4956, July 2007, <https://www.rfc-editor.org/info/rfc4956>.

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

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[RFC5893] Alvestrand, H., Ed. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.

[RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, DOI 10.17487/RFC6055, February 2011, <https://www.rfc-editor.org/info/rfc6055>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <https://www.rfc-editor.org/info/rfc6303>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <https://www.rfc-editor.org/info/rfc6365>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <https://www.rfc-editor.org/info/rfc7129>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

[RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin, "Inventory and Analysis of WHOIS Registration Objects", RFC 7485, DOI 10.17487/RFC7485, March 2015, <https://www.rfc-editor.org/info/rfc7485>.

[RFC8109] Koch, P., Larson, M., and P. Hoffman, "Initializing a DNS Resolver with Priming Queries", BCP 209, RFC 8109, DOI 10.17487/RFC8109, March 2017, <https://www.rfc-editor.org/info/rfc8109>.

[RSSAC026] Root Server System Advisory Committee (RSSAC), "RSSAC Lexicon", 2017, <https://www.icann.org/en/system/files/files/rssac-026-14mar17-en.pdf>.

Appendix A. Definitions Updated by this Document

The following definitions from RFCs are updated by this document:

- Forwarder in [RFC2308]
- Secure Entry Point (SEP) in [RFC3757]; note, however, that this RFC is already obsolete

Appendix B. Definitions First Defined in this Document

The following definitions are first defined in this document:

- "Alias" in Section 2
- "Apex" in Section 6
- "arpa" in Section 6
- "Class independent" in Section 4
- "Delegation-centric zone" in Section 6
- "Delegation" in Section 6
- "DNS operator" in Section 7
- "DNSSEC-aware" in Section 8
- "DNSSEC-unaware" in Section 8
- "Forwarding" in Section 5
- "Full resolver" in Section 5
- "Fully qualified domain name" in Section 2
- "Global DNS" in Section 2
- "Hardware Security Module (HSM)" in Section 8
- "Host name" in Section 2
- "IDN" in Section 2
- "In-bailiwick" in Section 6
- "Label" in Section 2
- "Locally served DNS zone" in Section 2
- "Naming system" in Section 2
- "Negative response" in Section 3
- "Open resolver" in Section 5

- "Out-of-bailiwick" in Section 6
- "Passive DNS" in Section 5
- "Policy-implementing resolver" in Section 5
- "Presentation format" in Section 4
- "Priming" in Section 5
- "Private DNS" in Section 2
- "Recursive resolver" in Section 5
- "Referrals" in Section 3
- "Registrant" in Section 7
- "Registrar" in Section 7
- "Registry" in Section 7
- "Root zone" in Section 6
- "Secure Entry Point (SEP)" in Section 8
- "Signing software" in Section 8
- "Stub resolver" in Section 5
- "TLD" in Section 2
- "Validating resolver" in Section 8
- "Validation" in Section 8
- "View" in Section 5
- "Zone transfer" in Section 5

Index

Acknowledgements

The following is the Acknowledgements for RFC 7719. Additional acknowledgements may be added as this draft is worked on.

The authors gratefully acknowledge all of the authors of DNS-related RFCs that proceed this one. Comments from Tony Finch, Stephane Bortzmeyer, Niall O'Reilly, Colm MacCarthaigh, Ray Bellis, John Kristoff, Robert Edmonds, Paul Wouters, Shumon Huque, Paul Ebersman, David Lawrence, Matthijs Mekking, Casey Deccio, Bob Harold, Ed Lewis, John Klensin, David Black, and many others in the DNSOP Working Group helped shape RFC 7719.

Additional people contributed to this document, including: John Dickinson, Bob Harold, Peter Koch, [[MORE NAMES WILL APPEAR HERE AS FOLKS CONTRIBUTE]].

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Andrew Sullivan
Oracle
100 Milverton Drive
Mississauga, ON L5R 4H1
Canada

Email: andrew.s.sullivan@oracle.com

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

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

Network Working Group
Internet-Draft
Obsoletes: 7719 (if approved)
Updates: 2308 (if approved)
Intended status: Best Current Practice
Expires: March 17, 2019

P. Hoffman
ICANN
A. Sullivan

K. Fujiwara
JPRS

September 13, 2018

DNS Terminology
draft-ietf-dnsop-terminology-bis-14

Abstract

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

This document obsoletes RFC 7719 and updates RFC 2308.

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 March 17, 2019.

Table of Contents

1. Introduction

The Domain Name System (DNS) is a simple query-response protocol whose messages in both directions have the same format. (Section 2 gives a definition of "public DNS", which is often what people mean when they say "the DNS".) The protocol and message format are defined in [RFC1034] and [RFC1035]. These RFCs defined some terms, but later documents defined others. Some of the terms from [RFC1034] and [RFC1035] now have somewhat different meanings than they did in 1987.

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

Most of the definitions here are the consensus definition of the DNS community -- both protocol developers and operators. Some of the definitions differ from earlier RFCs, and those differences are noted. In this document, where the consensus definition is the same as the one in an RFC, that RFC is quoted. Where the consensus definition has changed somewhat, the RFC is mentioned but the new stand-alone definition is given. See Appendix A for a list of the definitions that this document updates.

It is important to note that, during the development of this document, it became clear that some DNS-related terms are interpreted quite differently by different DNS experts. Further, some terms that are defined in early DNS RFCs now have definitions that are generally agreed to, but that are different from the original definitions. Therefore, this document is a substantial revision to [RFC7719].

The terms are organized loosely by topic. Some definitions are for new terms for things that are commonly talked about in the DNS community but that never had terms defined for them.

Other organizations sometimes define DNS-related terms their own way. For example, the WHATWG defines "domain" at <https://url.spec.whatwg.org/>. The Root Server System Advisory Committee (RSSAC) has a good lexicon [RSSAC026].

Note that there is no single consistent definition of "the DNS". It can be considered to be some combination of the following: a commonly used naming scheme for objects on the Internet; a distributed database representing the names and certain properties of these objects; an architecture providing distributed maintenance, resilience, and loose coherency for this database; and a simple query-response protocol (as mentioned below) implementing this architecture. Section 2 defines "global DNS" and "private DNS" as a way to deal with these differing definitions.

Capitalization in DNS terms is often inconsistent among RFCs and various DNS practitioners. The capitalization used in this document is a best guess at current practices, and is not meant to indicate that other capitalization styles are wrong or archaic. In some cases, multiple styles of capitalization are used for the same term due to quoting from different RFCs.

Readers should note that the terms in this document are grouped by topic. Someone who is not already familiar with the DNS can probably not learn about the DNS from scratch by reading this document from front to back. Instead, skipping around may be the only way to get enough context to understand some of the definitions. This document

has an index that might be useful for readers who are attempting to learn the DNS by reading this document.

2. Names

Naming system: A naming system associates names with data. Naming systems have many significant facets that help differentiate them from each other. Some commonly-identified facets include:

* Composition of names
* Format of names
* Administration of names
* Types of data that can be associated with names
* Types of metadata for names
* Protocol for getting data from a name
* Context for resolving a name

Note that this list is a small subset of facets that people have identified over time for naming systems, and the IETF has yet to agree on a good set of facets that can be used to compare naming systems. For example, other facets might include "protocol to update data in a name", "privacy of names", and "privacy of data associated with names", but those are not as well-defined as the ones listed above. The list here is chosen because it helps describe the DNS and naming systems similar to the DNS.

Domain name: An ordered list of one or more labels.

Note that this is a definition independent of the DNS RFCs, and the definition here also applies to systems other than the DNS. [RFC1034] defines the "domain name space" using mathematical trees and their nodes in graph theory, and this definition has the same practical result as the definition here. Any path of a directed acyclic graph can be represented by a domain name consisting of the labels of its nodes, ordered by decreasing distance from the root(s) (which is the normal convention within the DNS, including this document). A domain name whose last label identifies a root of the graph is fully qualified; other domain names whose labels form a strict prefix of a fully qualified domain name are relative to its first omitted node.

Also note that different IETF and non-IETF documents have used the term "domain name" in many different ways. It is common for earlier documents to use "domain name" to mean "names that match the syntax in [RFC1035]", but possibly with additional rules such as "and are, or will be, resolvable in the global DNS" or "but only using the presentation format".

Label: An ordered list of zero or more octets that makes up a portion of a domain name. Using graph theory, a label identifies one node in a portion of the graph of all possible domain names.

Global DNS: Using the short set of facets listed in "Naming system", the global DNS can be defined as follows. Most of the rules here come from [RFC1034] and [RFC1035], although the term "global DNS" has not been defined before now.

Composition of names -- A name in the global DNS has one or more labels. The length of each label is between 0 and 63 octets inclusive. In a fully-qualified domain name, the last label in the ordered list is 0 octets long; it is the only label whose length may be 0 octets, and it is called the "root" or "root label". A domain name in the global DNS has a maximum total length of 255 octets in the wire format; the root represents one octet for this calculation. (Multicast DNS [RFC6762] allows names up to 255 bytes plus a terminating zero byte based on a different interpretation of RFC 1035 and what is included in the 255 octets.)

Format of names -- Names in the global DNS are domain names. There are three formats: wire format, presentation format, and common display.

The basic wire format for names in the global DNS is a list of labels ordered by decreasing distance from the root, with the root label last. Each label is preceded by a length octet. [RFC1035] also defines a compression scheme that modifies this format.

The presentation format for names in the global DNS is a list of labels ordered by decreasing distance from the root, encoded as ASCII, with a "." character between each label. In presentation format, a fully-qualified domain name includes the root label and the associated separator dot. For example, in presentation format, a fully-qualified domain name with two non-root labels is always shown as "example.tld." instead of "example.tld". [RFC1035] defines a method for showing octets that do not display in ASCII.

The common display format is used in applications and free text. It is the same as the presentation format, but showing the root label and the "." before it is optional and is rarely done. For example, in common display format, a fully-qualified domain name with two non-root labels is usually shown as "example.tld" instead of "example.tld.". Names in the common display format are normally written such that the directionality of the writing system presents labels by decreasing distance from the root (so, in both English and the C programming language the root or TLD label in the ordered list is right-most; but in Arabic it may be left-most, depending on local conventions).

Administration of names -- Administration is specified by delegation (see the definition of "delegation" in Section 7). Policies for administration of the root zone in the global DNS are determined by the names operational community, which convenes itself in the Internet Corporation for Assigned Names and Numbers (ICANN). The names operational community selects the IANA Functions Operator for the global DNS root zone. At the time this document is published, that operator is Public Technical Identifiers (PTI). (See <https://pti.icann.org/> for more information about PTI operating the IANA Functions.) The name servers that serve the root zone are provided by independent root operators. Other zones in the global DNS have their own policies for administration.

Types of data that can be associated with names -- A name can have zero or more resource records associated with it. There are numerous types of resource records with unique data structures defined in many different RFCs and in the IANA registry at [IANA_Resource_Registry].

Types of metadata for names -- Any name that is published in the DNS appears as a set of resource records (see the definition of "RRset" in Section 5). Some names do not themselves have data associated with them in the DNS, but "appear" in the DNS anyway because they form part of a longer name that does have data associated with it (see the definition of "empty non-terminals" in Section 7).

Protocol for getting data from a name -- The protocol described in [RFC1035].

Context for resolving a name -- The global DNS root zone distributed by PTI.

Private DNS: Names that use the protocol described in [RFC1035] but that do not rely on the global DNS root zone, or names that are

otherwise not generally available on the Internet but are using the protocol described in [RFC1035]. A system can use both the global DNS and one or more private DNS systems; for example, see "Split DNS" in Section 6.

Note that domain names that do not appear in the DNS, and that are intended never to be looked up using the DNS protocol, are not part of the global DNS or a private DNS even though they are domain names.

Multicast DNS: "Multicast DNS (mDNS) provides the ability to perform DNS-like operations on the local link in the absence of any conventional Unicast DNS server. In addition, Multicast DNS designates a portion of the DNS namespace to be free for local use, without the need to pay any annual fee, and without the need to set up delegations or otherwise configure a conventional DNS server to answer for those names." (Quoted from [RFC6762], Abstract) Although it uses a compatible wire format, mDNS is strictly speaking a different protocol than DNS. Also, where the above quote says "a portion of the DNS namespace", it would be clearer to say "a portion of the domain name space" The names in mDNS are not intended to be looked up in the DNS.

Locally served DNS zone: A locally served DNS zone is a special case of private DNS. Names are resolved using the DNS protocol in a local context. [RFC6303] defines subdomains of IN-ADDR.ARPA that are locally served zones. Resolution of names through locally served zones may result in ambiguous results. For example, the same name may resolve to different results in different locally served DNS zone contexts. The context for a locally served DNS zone may be explicit, for example, as defined in [RFC6303], or implicit, as defined by local DNS administration and not known to the resolution client.

Fully qualified domain name (FQDN): This is often just a clear way of saying the same thing as "domain name of a node", as outlined above. However, the term is ambiguous. Strictly speaking, a fully qualified domain name would include every label, including the zero-length label of the root: such a name would be written "www.example.net." (note the terminating dot). But because every name eventually shares the common root, names are often written relative to the root (such as "www.example.net") and are still called "fully qualified". This term first appeared in [RFC0819]. In this document, names are often written relative to the root.

The need for the term "fully qualified domain name" comes from the existence of partially qualified domain names, which are names where one or more of the last labels in the ordered list are

omitted (for example, a domain name of "www" relative to "example.net" identifies "www.example.net"). Such relative names are understood only by context.

Host name: This term and its equivalent, "hostname", have been widely used but are not defined in [RFC1034], [RFC1035], [RFC1123], or [RFC2181]. The DNS was originally deployed into the Host Tables environment as outlined in [RFC0952], and it is likely that the term followed informally from the definition there. Over time, the definition seems to have shifted. "Host name" is often meant to be a domain name that follows the rules in Section 3.5 of [RFC1034], the "preferred name syntax" (that is, every character in each label is a letter, a digit, or a hyphen). Note that any label in a domain name can contain any octet value; hostnames are generally considered to be domain names where every label follows the rules in the "preferred name syntax", with the amendment that labels can start with ASCII digits (this amendment comes from Section 2.1 of [RFC1123]).

People also sometimes use the term hostname to refer to just the first label of an FQDN, such as "printer" in "printer.admin.example.com". (Sometimes this is formalized in configuration in operating systems.) In addition, people sometimes use this term to describe any name that refers to a machine, and those might include labels that do not conform to the "preferred name syntax".

TLD: A Top-Level Domain, meaning a zone that is one layer below the root, such as "com" or "jp". There is nothing special, from the point of view of the DNS, about TLDs. Most of them are also delegation-centric zones (defined in Section 7, and there are significant policy issues around their operation. TLDs are often divided into sub-groups such as Country Code Top-Level Domains (ccTLDs), Generic Top-Level Domains (gTLDs), and others; the division is a matter of policy, and beyond the scope of this document.

IDN: The common abbreviation for "Internationalized Domain Name". The IDNA protocol is the standard mechanism for handling domain names with non-ASCII characters in applications in the DNS. The current standard at the time of this writing, normally called "IDNA2008", is defined in [RFC5890], [RFC5891], [RFC5892], [RFC5893], and [RFC5894]. These documents define many IDN-specific terms such as "LDH label", "A-label", and "U-label". [RFC6365] defines more terms that relate to internationalization (some of which relate to IDNs), and [RFC6055] has a much more extensive discussion of IDNs, including some new terminology.

Subdomain: "A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name." (Quoted from [RFC1034], Section 3.1). For example, in the host name "nnn.mmm.example.com", both "mmm.example.com" and "nnn.mmm.example.com" are subdomains of "example.com". Note that the comparisons here are done on whole labels; that is, "ooo.example.com" is not a subdomain of "oo.example.com".

Alias: The owner of a CNAME resource record, or a subdomain of the owner of a DNAME resource record (DNAME records are defined in [RFC6672]). See also "canonical name".

Canonical name: A CNAME resource record "identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR." (Quoted from [RFC1034], Section 3.6.2) This usage of the word "canonical" is related to the mathematical concept of "canonical form".

CNAME: "It is traditional to refer to the owner of a CNAME record as 'a CNAME'. This is unfortunate, as 'CNAME' is an abbreviation of 'canonical name', and the owner of a CNAME record is an alias, not a canonical name." (Quoted from [RFC2181], Section 10.1.1)

3. DNS Response Codes

Some of response codes that are defined in [RFC1035] have acquired their own shorthand names. All of the RCODEs are listed at [IANA_Resource_Registry], although that site uses mixed-case capitalization, while most documents use all-caps. Some of the common names are described here, but the official list is in the IANA registry.

NOERROR: "No error condition" (Quoted from [RFC1035], Section 4.1.1.)

FORMERR: "Format error - The name server was unable to interpret the query." (Quoted from [RFC1035], Section 4.1.1.)

SERVFAIL: "Server failure - The name server was unable to process this query due to a problem with the name server." (Quoted from [RFC1035], Section 4.1.1.)

NXDOMAIN: "Name Error - This code signifies that the domain name referenced in the query does not exist." (Quoted from [RFC1035], Section 4.1.1.) [RFC2308] established NXDOMAIN as a synonym for Name Error.

NOTIMP: "Not Implemented - The name server does not support the requested kind of query." (Quoted from [RFC1035], Section 4.1.1.)

REFUSED: "Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data." (Quoted from [RFC1035], Section 4.1.1.)

NODATA: "A pseudo RCODE which indicates that the name is valid for the given class, but there are no records of the given type. A NODATA response has to be inferred from the answer." (Quoted from [RFC2308], Section 1.) "NODATA is indicated by an answer with the RCODE set to NOERROR and no relevant answers in the answer section. The authority section will contain an SOA record, or there will be no NS records there." (Quoted from [RFC2308], Section 2.2.) Note that referrals have a similar format to NODATA replies; [RFC2308] explains how to distinguish them.

The term "NXRRSET" is sometimes used as a synonym for NODATA. However, this is a mistake, given that NXRRSET is a specific error code defined in [RFC2136].

Negative response: A response that indicates that a particular RRset does not exist, or whose RCODE indicates the nameserver cannot answer. Sections 2 and 7 of [RFC2308] describe the types of negative responses in detail.

4. DNS Transactions

The header of a DNS message is its first 12 octets. Many of the fields and flags in the header diagram in Sections 4.1.1 through 4.1.3 of [RFC1035] are referred to by their names in that diagram. For example, the response codes are called "RCODEs", the data for a record is called the "RDATA", and the authoritative answer bit is often called "the AA flag" or "the AA bit".

Class: A class "identifies a protocol family or instance of a protocol" (Quoted from [RFC1034], Section 3.6). "The DNS tags all data with a class as well as the type, so that we can allow parallel use of different formats for data of type address." (Quoted from [RFC1034], Section 2.2). In practice, the class for nearly every query is "IN". There are some queries for "CH", but they are usually for the purposes of information about the server itself rather than for a different type of address.

QNAME: The most commonly-used rough definition is that the QNAME is a field in the Question section of a query. "A standard query specifies a target domain name (QNAME), query type (QTYPE), and query class (QCLASS) and asks for RRs which match." (Quoted from [RFC1034], Section 3.7.1.). Strictly speaking, the definition comes from [RFC1035], Section 4.1.2, where the QNAME is defined in respect of the Question Section. This definition appears to be applied consistently: the discussion of inverse queries in section 6.4 refers to the "owner name of the query RR and its TTL", because inverse queries populate the Answer Section and leave the Question Section empty. (Inverse queries are deprecated in [RFC3425], and so relevant definitions do not appear in this document.)

[RFC2308], however, has an alternate definition that puts the QNAME in the answer (or series of answers) instead of the query. It defines QNAME as: "...the name in the query section of an answer, or where this resolves to a CNAME, or CNAME chain, the data field of the last CNAME. The last CNAME in this sense is that which contains a value which does not resolve to another CNAME." This definition has a certain internal logic, because of the way CNAME substitution works and the definition of CNAME. If a name server does not find an RRset that matches a query, but it finds the same name in the same class with a CNAME record, then the name server "includes the CNAME record in the response and restarts the query at the domain name specified in the data field of the CNAME record." (Quoted from [RFC1034] Section 3.6.2). This is made explicit in the resolution algorithm outlined in Section 4.3.2 of [RFC1034], which says to "change QNAME to the canonical name in the CNAME RR, and go back to step 1" in the case of a CNAME RR. Since a CNAME record explicitly declares that the owner name is canonically named what is in the RDATA, then there is a way to view the new name (i.e. the name that was in the RDATA of the CNAME RR) as also being the QNAME.

This creates a kind of confusion, however, because the response to a query that results in CNAME processing contains in the echoed Question Section one QNAME (the name in the original query), and a second QNAME that is in the data field of the last CNAME. The confusion comes from the iterative/recursive mode of resolution, which finally returns an answer that need not actually have the same owner name as the QNAME contained in the original query.

To address this potential confusion, it is helpful to distinguish between three meanings:

* QNAME (original): The name actually sent in the Question Section in the original query, which is always echoed in the

(final) reply in the Question Section when the QR bit is set to 1.

* QNAME (effective): A name actually resolved, which is either the name originally queried, or a name received in a CNAME chain response.

* QNAME (final): The name actually resolved, which is either the name actually queried or else the last name in a CNAME chain response.

Note that, because the definition in [RFC2308] is actually for a different concept than what was in [RFC1034], it would have been better if [RFC2308] had used a different name for that concept. In general use today, QNAME almost always means what is defined above as "QNAME (original)".

Referrals: A type of response in which a server, signaling that it is not (completely) authoritative for an answer, provides the querying resolver with an alternative place to send its query. Referrals can be partial.

A referral arises when a server is not performing recursive service while answering a query. It appears in step 3(b) of the algorithm in [RFC1034], Section 4.3.2.

There are two types of referral response. The first is a downward referral (sometimes described as "delegation response"), where the server is authoritative for some portion of the QNAME. The authority section RRset's RDATA contains the name servers specified at the referred-to zone cut. In normal DNS operation, this kind of response is required in order to find names beneath a delegation. The bare use of "referral" means this kind of referral, and many people believe that this is the only legitimate kind of referral in the DNS.

The second is an upward referral (sometimes described as "root referral"), where the server is not authoritative for any portion of the QNAME. When this happens, the referred-to zone in the authority section is usually the root zone (.). In normal DNS operation, this kind of response is not required for resolution or for correctly answering any query. There is no requirement that any server send upward referrals. Some people regard upward referrals as a sign of a misconfiguration or error. Upward referrals always need some sort of qualifier (such as "upward" or "root"), and are never identified by the bare word "referral".

A response that has only a referral contains an empty answer section. It contains the NS RRset for the referred-to zone in the authority section. It may contain RRs that provide addresses in the additional section. The AA bit is clear.

In the case where the query matches an alias, and the server is not authoritative for the target of the alias but it is authoritative for some name above the target of the alias, the resolution algorithm will produce a response that contains both the authoritative answer for the alias, and also a referral. Such a partial answer and referral response has data in the answer section. It has the NS RRset for the referred-to zone in the authority section. It may contain RRs that provide addresses in the additional section. The AA bit is set, because the first name in the answer section matches the QNAME and the server is authoritative for that answer (see [RFC1035], Section 4.1.1).

5. Resource Records

RR: An acronym for resource record. ([RFC1034], Section 3.6.)

RRset: A set of resource records "with the same label, class and type, but with different data". (Definition from [RFC2181], Section 5) Also spelled RRSet in some documents. As a clarification, "same label" in this definition means "same owner name". In addition, [RFC2181] states that "the TTLs of all RRs in an RRSet must be the same".

Note that RRSIG resource records do not match this definition. [RFC4035] says: "An RRset MAY have multiple RRSIG RRs associated with it. Note that as RRSIG RRs are closely tied to the RRsets whose signatures they contain, RRSIG RRs, unlike all other DNS RR types, do not form RRsets. In particular, the TTL values among RRSIG RRs with a common owner name do not follow the RRset rules described in [RFC2181]."

Master file: "Master files are text files that contain RRs in text form. Since the contents of a zone can be expressed in the form of a list of RRs a master file is most often used to define a zone, though it can be used to list a cache’s contents." (Quoted from [RFC1035], Section 5.) Master files are sometimes called "zone files".

Presentation format: The text format used in master files. This format is shown but not formally defined in [RFC1034] and [RFC1035]. The term "presentation format" first appears in [RFC4034].

EDNS: The extension mechanisms for DNS, defined in [RFC6891]. Sometimes called "EDNS0" or "EDNS(0)" to indicate the version number. EDNS allows DNS clients and servers to specify message sizes larger than the original 512 octet limit, to expand the response code space, and to carry additional options that affect the handling of a DNS query.

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

Owner: "The domain name where a RR is found" (Quoted from [RFC1034], Section 3.6). Often appears in the term "owner name".

SOA field names: DNS documents, including the definitions here, often refer to the fields in the RDATA of an SOA resource record by field name. "SOA" stands for "start of a zone of authority". Those fields are defined in Section 3.3.13 of [RFC1035]. The names (in the order they appear in the SOA RDATA) are MNAME, RNAME, SERIAL, REFRESH, RETRY, EXPIRE, and MINIMUM. Note that the meaning of MINIMUM field is updated in Section 4 of [RFC2308]; the new definition is that the MINIMUM field is only "the TTL to be used for negative responses". This document tends to use field names instead of terms that describe the fields.

TTL: The maximum "time to live" of a resource record. "A TTL value is an unsigned number, with a minimum value of 0, and a maximum value of 2147483647. That is, a maximum of 2^31 - 1. When transmitted, the TTL is encoded in the less significant 31 bits of the 32 bit TTL field, with the most significant, or sign, bit set to zero." (Quoted from [RFC2181], Section 8) (Note that [RFC1035] erroneously stated that this is a signed integer; that was fixed by [RFC2181].)

The TTL "specifies the time interval that the resource record may be cached before the source of the information should again be consulted". (Quoted from [RFC1035], Section 3.2.1) Also: "the time interval (in seconds) that the resource record may be cached before it should be discarded". (Quoted from [RFC1035], Section 4.1.3). Despite being defined for a resource record, the TTL of every resource record in an RRset is required to be the same ([RFC2181], Section 5.2).

The reason that the TTL is the maximum time to live is that a cache operator might decide to shorten the time to live for operational purposes, such as if there is a policy to disallow TTL values over a certain number. Some servers are known to ignore

the TTL on some RRsets (such as when the authoritative data has a very short TTL) even though this is against the advice in RFC 1035. An RRset can be flushed from the cache before the end of the TTL interval, at which point the value of the TTL becomes unknown because the RRset with which it was associated no longer exists.

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

Class independent: A resource record type whose syntax and semantics are the same for every DNS class. A resource record type that is not class independent has different meanings depending on the DNS class of the record, or the meaning is undefined for some class. Most resource record types are defined for class 1 (IN, the Internet), but many are undefined for other classes.

Address records: Records whose type is A or AAAA. [RFC2181] informally defines these as "(A, AAAA, etc)". Note that new types of address records could be defined in the future.

6. DNS Servers and Clients

This section defines the terms used for the systems that act as DNS clients, DNS servers, or both. In the RFCs, DNS servers are sometimes called "name servers", "nameservers", or just "servers". There is no formal definition of DNS server, but the RFCs generally assume that it is an Internet server that listens for queries and sends responses using the DNS protocol defined in [RFC1035] and its successors.

It is important to note that the terms "DNS server" and "name server" require context in order to understand the services being provided. Both authoritative servers and recursive resolvers are often called "DNS servers" and "name servers" even though they serve different roles (but may be part of the same software package).

For terminology specific to the public DNS root server system, see [RSSAC026]. That document defines terms such as "root server", "root server operator", and terms that are specific to the way that the root zone of the public DNS is served.

Resolver: A program "that extract[s] information from name servers in response to client requests." (Quoted from [RFC1034], Section 2.4) A resolver performs queries for a name, type, and

class, and receives responses. The logical function is called "resolution". In practice, the term is usually referring to some specific type of resolver (some of which are defined below), and understanding the use of the term depends on understanding the context.

A related term is "resolve", which is not formally defined in [RFC1034] or [RFC1035]. An imputed definition might be "asking a question that consists of a domain name, class, and type, and receiving some sort of response". Similarly, an imputed definition of "resolution" might be "the response received from resolving".

Stub resolver: A resolver that cannot perform all resolution itself. Stub resolvers generally depend on a recursive resolver to undertake the actual resolution function. Stub resolvers are discussed but never fully defined in Section 5.3.1 of [RFC1034]. They are fully defined in Section 6.1.3.1 of [RFC1123].

Iterative mode: A resolution mode of a server that receives DNS queries and responds with a referral to another server. Section 2.3 of [RFC1034] describes this as "The server refers the client to another server and lets the client pursue the query". A resolver that works in iterative mode is sometimes called an "iterative resolver". See also "iterative resolution" later in this section.

Recursive mode: A resolution mode of a server that receives DNS queries and either responds to those queries from a local cache or sends queries to other servers in order to get the final answers to the original queries. Section 2.3 of [RFC1034] describes this as "The first server pursues the query for the client at another server". Section 4.3.1 of [RFC1034] says "in [recursive] mode the name server acts in the role of a resolver and returns either an error or the answer, but never referrals." That same section also says "The recursive mode occurs when a query with RD set arrives at a server which is willing to provide recursive service; the client can verify that recursive mode was used by checking that both RA and RD are set in the reply."

A server operating in recursive mode may be thought of as having a name server side (which is what answers the query) and a resolver side (which performs the resolution function). Systems operating in this mode are commonly called "recursive servers". Sometimes they are called "recursive resolvers". In practice it is not possible to know in advance whether the server that one is querying will also perform recursion; both terms can be observed in use interchangeably.

Recursive resolver: A resolver that acts in recursive mode. In general, a recursive resolver is expected to cache the answers it receives (which would make it a full-service resolver), but some recursive resolvers might not cache.

[RFC4697] tried to differentiate between a recursive resolver and an iterative resolver.

Recursive query: A query with the Recursion Desired (RD) bit set to 1 in the header. (See Section 4.1.1 of [RFC1035].) If recursive service is available and is requested by the RD bit in the query, the server uses its resolver to answer the query. (See Section 4.3.2 of [RFC1035].)

Non-recursive query: A query with the Recursion Desired (RD) bit set to 0 in the header. A server can answer non-recursive queries using only local information: the response contains either an error, the answer, or a referral to some other server "closer" to the answer. (See Section 4.3.1 of [RFC1035].)

Iterative resolution: A name server may be presented with a query that can only be answered by some other server. The two general approaches to dealing with this problem are "recursive", in which the first server pursues the query on behalf of the client at another server, and "iterative", in which the server refers the client to another server and lets the client pursue the query there. (See Section 2.3 of [RFC1034].)

In iterative resolution, the client repeatedly makes non-recursive queries and follows referrals and/or aliases. The iterative resolution algorithm is described in Section 5.3.3 of [RFC1034].

Full resolver: This term is used in [RFC1035], but it is not defined there. RFC 1123 defines a "full-service resolver" that may or may not be what was intended by "full resolver" in [RFC1035]. This term is not properly defined in any RFC.

Full-service resolver: Section 6.1.3.1 of [RFC1123] defines this term to mean a resolver that acts in recursive mode with a cache (and meets other requirements).

Priming: "The act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers." (Quoted from [RFC8109], Section 2.) In order to operate in recursive mode, a resolver needs to know the address of at least one root server. Priming is most often done from a configuration setting that contains a list of authoritative servers for the root zone.

Root hints: "Operators who manage a DNS recursive resolver typically need to configure a 'root hints file'. This file contains the names and IP addresses of the authoritative name servers for the root zone, so the software can bootstrap the DNS resolution process. For many pieces of software, this list comes built into the software." (Quoted from [IANA_RootFiles]) This file is often used in priming.

Negative caching: "The storage of knowledge that something does not exist, cannot give an answer, or does not give an answer." (Quoted from [RFC2308], Section 1)

Authoritative server: "A server that knows the content of a DNS zone from local knowledge, and thus can answer queries about that zone without needing to query other servers." (Quoted from [RFC2182], Section 2.) An authoritative server is named in the NS ("name server") record in a zone. It is a system that responds to DNS queries with information about zones for which it has been configured to answer with the AA flag in the response header set to 1. It is a server that has authority over one or more DNS zones. Note that it is possible for an authoritative server to respond to a query without the parent zone delegating authority to that server. Authoritative servers also provide "referrals", usually to child zones delegated from them; these referrals have the AA bit set to 0 and come with referral data in the Authority and (if needed) the Additional sections.

Authoritative-only server: A name server that only serves authoritative data and ignores requests for recursion. It will "not normally generate any queries of its own. Instead, it answers non-recursive queries from iterative resolvers looking for information in zones it serves." (Quoted from [RFC4697], Section 2.4) In this case, "ignores requests for recursion" means "responds to requests for recursion with responses indicating that recursion was not performed".

Zone transfer: The act of a client requesting a copy of a zone and an authoritative server sending the needed information. (See Section 7 for a description of zones.) There are two common standard ways to do zone transfers: the AXFR ("Authoritative Transfer") mechanism to copy the full zone (described in [RFC5936], and the IXFR ("Incremental Transfer") mechanism to copy only parts of the zone that have changed (described in [RFC1995]). Many systems use non-standard methods for zone transfer outside the DNS protocol.

Slave server: See "Secondary server".

Secondary server: "An authoritative server which uses zone transfer to retrieve the zone" (Quoted from [RFC1996], Section 2.1). Secondary servers are also discussed in [RFC1034]. [RFC2182] describes secondary servers in more detail. Although early DNS RFCs such as [RFC1996] referred to this as a "slave", the current common usage has shifted to calling it a "secondary".

Master server: See "Primary server".

Primary server: "Any authoritative server configured to be the source of zone transfer for one or more [secondary] servers" (Quoted from [RFC1996], Section 2.1) or, more specifically, "an authoritative server configured to be the source of AXFR or IXFR data for one or more [secondary] servers" (Quoted from [RFC2136]). Primary servers are also discussed in [RFC1034]. Although early DNS RFCs such as [RFC1996] referred to this as a "master", the current common usage has shifted to "primary".

Primary master: "The primary master is named in the zone's SOA MNAME field and optionally by an NS RR". (Quoted from [RFC1996], Section 2.1). [RFC2136] defines "primary master" as "Master server at the root of the AXFR/IXFR dependency graph. The primary master is named in the zone's SOA MNAME field and optionally by an NS RR. There is by definition only one primary master server per zone."

The idea of a primary master is only used in [RFC1996] and [RFC2136]. A modern interpretation of the term "primary master" is a server that is both authoritative for a zone and that gets its updates to the zone from configuration (such as a master file) or from UPDATE transactions.

Stealth server: This is "like a slave server except not listed in an NS RR for the zone." (Quoted from [RFC1996], Section 2.1)

Hidden master: A stealth server that is a primary server for zone transfers. "In this arrangement, the master name server that processes the updates is unavailable to general hosts on the Internet; it is not listed in the NS RRset." (Quoted from [RFC6781], Section 3.4.3). An earlier RFC, [RFC4641], said that the hidden master's name "appears in the SOA RRs MNAME field", although in some setups, the name does not appear at all in the public DNS. A hidden master can also be a secondary server for the zone itself.

Forwarding: The process of one server sending a DNS query with the RD bit set to 1 to another server to resolve that query.

Forwarding is a function of a DNS resolver; it is different than simply blindly relaying queries.

[RFC5625] does not give a specific definition for forwarding, but describes in detail what features a system that forwards needs to support. Systems that forward are sometimes called "DNS proxies", but that term has not yet been defined (even in [RFC5625]).

Forwarder: Section 1 of [RFC2308] describes a forwarder as "a nameserver used to resolve queries instead of directly using the authoritative nameserver chain". [RFC2308] further says "The forwarder typically either has better access to the internet, or maintains a bigger cache which may be shared amongst many resolvers." That definition appears to suggest that forwarders normally only query authoritative servers. In current use, however, forwarders often stand between stub resolvers and recursive servers. [RFC2308] is silent on whether a forwarder is iterative-only or can be a full-service resolver.

Policy-implementing resolver: A resolver acting in recursive mode that changes some of the answers that it returns based on policy criteria, such as to prevent access to malware sites or objectionable content. In general, a stub resolver has no idea whether upstream resolvers implement such policy or, if they do, the exact policy about what changes will be made. In some cases, the user of the stub resolver has selected the policy-implementing resolver with the explicit intention of using it to implement the policies. In other cases, policies are imposed without the user of the stub resolver being informed.

Open resolver: A full-service resolver that accepts and processes queries from any (or nearly any) client. This is sometimes also called a "public resolver", although the term "public resolver" is used more with open resolvers that are meant to be open, as compared to the vast majority of open resolvers that are probably misconfigured to be open. Open resolvers are discussed in [RFC5358]

Split DNS: The terms "split DNS" and "split-horizon DNS" have long been used in the DNS community without formal definition. In general, they refer to situations in which DNS servers that are authoritative for a particular set of domains provide partly or completely different answers in those domains depending on the source of the query. The effect of this is that a domain name that is notionally globally unique nevertheless has different meanings for different network users. This can sometimes be the result of a "view" configuration, described below.

[RFC2775], Section 3.8 gives a related definition that is too specific to be generally useful.

View: A configuration for a DNS server that allows it to provide different responses depending on attributes of the query, such as for "split DNS". Typically, views differ by the source IP address of a query, but can also be based on the destination IP address, the type of query (such as AXFR), whether it is recursive, and so on. Views are often used to provide more names or different addresses to queries from "inside" a protected network than to those "outside" that network. Views are not a standardized part of the DNS, but they are widely implemented in server software.

Passive DNS: A mechanism to collect DNS data by storing DNS responses from name servers. Some of these systems also collect the DNS queries associated with the responses, although doing so raises some privacy concerns. Passive DNS databases can be used to answer historical questions about DNS zones such as which values were present at a given time in the past, or when a name was spotted first. Passive DNS databases allow searching of the stored records on keys other than just the name and type, such as "find all names which have A records of a particular value".

Anycast: "The practice of making a particular service address available in multiple, discrete, autonomous locations, such that datagrams sent are routed to one of several available locations." (Quoted from [RFC4786], Section 2) See [RFC4786] for more detail on Anycast and other terms that are specific to its use.

Instance: "When anycast routing is used to allow more than one server to have the same IP address, each one of those servers is commonly referred to as an 'instance'." "An instance of a server, such as a root server, is often referred to as an 'Anycast instance'." (Quoted from [RSSAC026])

Privacy-enabling DNS server: "A DNS server that implements DNS over TLS [RFC7858] and may optionally implement DNS over DTLS [RFC8094]." (Quoted from [RFC8310], Section 2) Other types of DNS servers might also be considerd privacy-enabling, such as those running DNS over HTTPS [I-D.ietf-doh-dns-over-https].

7. Zones

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

Zone: "Authoritative information is organized into units called 'zones', and these zones can be automatically distributed to the

name servers which provide redundant service for the data in a zone." (Quoted from [RFC1034], Section 2.4)

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

Parent: "The domain in which the Child is registered." (Quoted from [RFC7344], Section 1.1) Earlier, "parent name server" was defined in [RFC0882] as "the name server that has authority over the place in the domain name space that will hold the new domain". (Note that [RFC0882] was obsoleted by [RFC1034] and [RFC1035].) [RFC0819] also has some description of the relationship between parents and children.

Origin:

There are two different uses for this term:

(a) "The domain name that appears at the top of a zone (just below the cut that separates the zone from its parent). The name of the zone is the same as the name of the domain at the zone's origin." (Quoted from [RFC2181], Section 6.) These days, this sense of "origin" and "apex" (defined below) are often used interchangeably.

(b) The domain name within which a given relative domain name appears in zone files. Generally seen in the context of "$ORIGIN", which is a control entry defined in [RFC1035], Section 5.1, as part of the master file format. For example, if the $ORIGIN is set to "example.org.", then a master file line for "www" is in fact an entry for "www.example.org.".

Apex: The point in the tree at an owner of an SOA and corresponding authoritative NS RRset. This is also called the "zone apex". [RFC4033] defines it as "the name at the child's side of a zone cut". The "apex" can usefully be thought of as a data-theoretic description of a tree structure, and "origin" is the name of the same concept when it is implemented in zone files. The distinction is not always maintained in use, however, and one can find uses that conflict subtly with this definition. [RFC1034] uses the term "top node of the zone" as a synonym of "apex", but that term is not widely used. These days, the first sense of "origin" (above) and "apex" are often used interchangeably.

Zone cut: The delimitation point between two zones where the origin of one of the zones is the child of the other zone.

"Zones are delimited by 'zone cuts'. Each zone cut separates a 'child' zone (below the cut) from a 'parent' zone (above the cut)." (Quoted from [RFC2181], Section 6; note that this is barely an ostensive definition.) Section 4.2 of [RFC1034] uses "cuts" instead of "zone cut".

Delegation: The process by which a separate zone is created in the name space beneath the apex of a given domain. Delegation happens when an NS RRset is added in the parent zone for the child origin. Delegation inherently happens at a zone cut. The term is also commonly a noun: the new zone that is created by the act of delegating.

Authoritative data: "All of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone." (Quoted from [RFC1034], Section 4.2.1) Note that this definition might inadvertently also cause any NS records that appear in the zone to be included, even those that might not truly be authoritative because there are identical NS RRs below the zone cut. This reveals the ambiguity in the notion of authoritative data, because the parent-side NS records authoritatively indicate the delegation, even though they are not themselves authoritative data.

[RFC4033], Section 2, defines "Authoritative RRset" which is related to authoritative data but has a more precise definition.

Lame delegation: "A lame delegations exists when a nameserver is delegated responsibility for providing nameservice for a zone (via NS records) but is not performing nameservice for that zone (usually because it is not set up as a primary or secondary for the zone)." (Quoted from [RFC1912], Section 2.8)

Another definition is that a lame delegation "happens when a name server is listed in the NS records for some domain and in fact it is not a server for that domain. Queries are thus sent to the wrong servers, who don't know nothing (at least not as expected) about the queried domain. Furthermore, sometimes these hosts (if they exist!) don't even run name servers." (Quoted from [RFC1713], Section 2.3)

Glue records: "[Resource records] which are not part of the authoritative data [of the zone], and are address resource records for the [name servers in subzones]. These RRs are only necessary if the name server's name is 'below' the cut, and are only used as part of a referral response." Without glue "we could be faced with the situation where the NS RRs tell us that in order to learn a name server's address, we should contact the server using the

address we wish to learn." (Definition from [RFC1034], Section 4.2.1)

A later definition is that glue "includes any record in a zone file that is not properly part of that zone, including nameserver records of delegated sub-zones (NS records), address records that accompany those NS records (A, AAAA, etc), and any other stray data that might appear" (Quoted from [RFC2181], Section 5.4.1). Although glue is sometimes used today with this wider definition in mind, the context surrounding the [RFC2181] definition suggests it is intended to apply to the use of glue within the document itself and not necessarily beyond.

Bailiwick: "In-bailiwick" is an adjective to describe a name server whose name is either a subdomain of or (rarely) the same as the origin of the zone that contains the delegation to the name server. In-bailiwick name servers may have glue records in their parent zone (using the first of the definitions of "glue records" in the definition above). (The term "bailiwick" means the district or territory where a bailiff or policeman has jurisdiction.)

"In-bailiwick" names are divided into two type of name server names: "in-domain" names and "sibling domain" names.

* In-domain: an adjective to describe a name server whose name is either subordinate to or (rarely) the same as the owner name of the NS resource records. An in-domain name server name MUST have glue records or name resolution fails. For example, a delegation for "child.example.com" may have "in-domain" name server name "ns.child.example.com".

* Sibling domain: a name server’s name that is either subordinate to or (rarely) the same as the zone origin and not subordinate to or the same as the owner name of the NS resource records. Glue records for sibling domains are allowed, but not necessary. For example, a delegation for "child.example.com" in "example.com" zone may have "sibling" name server name "ns.another.example.com".

"Out-of-bailiwick" is the antonym of in-bailiwick. An adjective to describe a name server whose name is not subordinate to or the same as the zone origin. Glue records for out-of-bailiwick name servers are useless. Following table shows examples of delegation types.

Delegation	Parent	Name Server Name	Type
com	.	a.gtld-servers.net	in-bailiwick / sibling domain
net	.	a.gtld-servers.net	in-bailiwick / in-domain
example.org	org	ns.example.org	in-bailiwick / in-domain
example.org	org	ns.ietf.org	in-bailiwick / sibling domain
example.org	org	ns.example.com	out-of-bailiwick
example.jp	jp	ns.example.jp	in-bailiwick / in-domain
example.jp	jp	ns.example.ne.jp	in-bailiwick / sibling domain
example.jp	jp	ns.example.com	out-of-bailiwick

Root zone: The zone of a DNS-based tree whose apex is the zero-length label. Also sometimes called "the DNS root".

Empty non-terminals (ENT): "Domain names that own no resource records but have subdomains that do." (Quoted from [RFC4592], Section 2.2.2.) A typical example is in SRV records: in the name "_sip._tcp.example.com", it is likely that "_tcp.example.com" has no RRsets, but that "_sip._tcp.example.com" has (at least) an SRV RRset.

Delegation-centric zone: A zone that consists mostly of delegations to child zones. This term is used in contrast to a zone that might have some delegations to child zones, but also has many data resource records for the zone itself and/or for child zones. The term is used in [RFC4956] and [RFC5155], but is not defined there.

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

Fast flux DNS: This "occurs when a domain is found in DNS using A records to multiple IP addresses, each of which has a very short Time-to-Live (TTL) value associated with it. This means that the domain resolves to varying IP addresses over a short period of time." (Quoted from [RFC6561], Section 1.1.5, with typo corrected) In addition to having legitimate uses, fast flux DNS can used to deliver malware. Because the addresses change so rapidly, it is difficult to ascertain all the hosts. It should be noted that the technique also works with AAAA records, but such use is not frequently observed on the Internet as of this writing.

Reverse DNS, reverse lookup: "The process of mapping an address to a name is generally known as a ’reverse lookup’, and the IN-

ADDR.ARPA and IP6.ARPA zones are said to support the 'reverse DNS'." (Quoted from [RFC5855], Section 1)

Forward lookup: "Hostname-to-address translation". (Quoted from [RFC2133], Section 6)

arpa: Address and Routing Parameter Area Domain: "The 'arpa' domain was originally established as part of the initial deployment of the DNS, to provide a transition mechanism from the Host Tables that were common in the ARPANET, as well as a home for the IPv4 reverse mapping domain. During 2000, the abbreviation was redesignated to 'Address and Routing Parameter Area' in the hope of reducing confusion with the earlier network name." (Quoted from [RFC3172], Section 2.) .arpa is an "infrastructure domain", a domain whose "role is to support the operating infrastructure of the Internet". (Quoted from [RFC3172], Section 2.) See [RFC3172] for more history of this name.

Service name: "Service names are the unique key in the Service Name and Transport Protocol Port Number registry. This unique symbolic name for a service may also be used for other purposes, such as in DNS SRV records." (Quoted from [RFC6335], Section 5.)

8. Wildcards

Wildcard: [RFC1034] defined "wildcard", but in a way that turned out to be confusing to implementers. For an extended discussion of wildcards, including clearer definitions, see [RFC4592]. Special treatment is given to RRs with owner names starting with the label "*". "Such RRs are called 'wildcards'. Wildcard RRs can be thought of as instructions for synthesizing RRs." (Quoted from [RFC1034], Section 4.3.3)

Asterisk label: "The first octet is the normal label type and length for a 1-octet-long label, and the second octet is the ASCII representation for the '*' character. A descriptive name of a label equaling that value is an 'asterisk label'." (Quoted from [RFC4592], Section 2.1.1)

Wildcard domain name: "A 'wildcard domain name' is defined by having its initial (i.e., leftmost or least significant) label be asterisk label." (Quoted from [RFC4592], Section 2.1.1)

Closest encloser: "The longest existing ancestor of a name." (Quoted from [RFC5155], Section 1.3) An earlier definition is "The node in the zone's tree of existing domain names that has the most labels matching the query name (consecutively, counting from the root label downward). Each match is a 'label match' and the order

of the labels is the same." (Quoted from [RFC4592], Section 3.3.1)

Closest provable encloser: "The longest ancestor of a name that can be proven to exist. Note that this is only different from the closest encloser in an Opt-Out zone." (Quoted from [RFC5155], Section 1.3) See Section 10 for more on "opt-out".

Next closer name: "The name one label longer than the closest provable encloser of a name." (Quoted from [RFC5155], Section 1.3)

Source of Synthesis: "The source of synthesis is defined in the context of a query process as that wildcard domain name immediately descending from the closest encloser, provided that this wildcard domain name exists. 'Immediately descending' means that the source of synthesis has a name of the form: <asterisk label>.<closest encloser>." (Quoted from [RFC4592], Section 3.3.1)

9. Registration Model

Registry: The administrative operation of a zone that allows registration of names within that zone. People often use this term to refer only to those organizations that perform registration in large delegation-centric zones (such as TLDs); but formally, whoever decides what data goes into a zone is the registry for that zone. This definition of "registry" is from a DNS point of view; for some zones, the policies that determine what can go in the zone are decided by zones that are superordinate and not the registry operator.

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

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

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

WHOIS: A protocol specified in [RFC3912], often used for querying registry databases. WHOIS data is frequently used to associate

registration data (such as zone management contacts) with domain names. The term "WHOIS data" is often used as a synonym for the registry database, even though that database may be served by different protocols, particularly RDAP. The WHOIS protocol is also used with IP address registry data.

RDAP: The Registration Data Access Protocol, defined in [RFC7480], [RFC7481], [RFC7482], [RFC7483], [RFC7484], and [RFC7485]. The RDAP protocol and data format are meant as a replacement for WHOIS.

DNS operator: An entity responsible for running DNS servers. For a zone's authoritative servers, the registrant may act as their own DNS operator, or their registrar may do it on their behalf, or they may use a third-party operator. For some zones, the registry function is performed by the DNS operator plus other entities who decide about the allowed contents of the zone.

Public suffix: "A domain that is controlled by a public registry." (Quoted from [RFC6265], Section 5.3) A common definition for this term is a domain under which subdomains can be registered by third parties, and on which HTTP cookies (which are described in detail in [RFC6265]) should not be set. There is no indication in a domain name whether it is a public suffix; that can only be determined by outside means. In fact, both a domain and a subdomain of that domain can be public suffixes.

There is nothing inherent in a domain name to indicate whether it is a public suffix. One resource for identifying public suffixes is the Public Suffix List (PSL) maintained by Mozilla (http://publicsuffix.org/).

For example, at the time this document is published, the "com.au" domain is listed as a public suffix in the PSL. (Note that this example might change in the future.)

Note that the term "public suffix" is controversial in the DNS community for many reasons, and may be significantly changed in the future. One example of the difficulty of calling a domain a public suffix is that designation can change over time as the registration policy for the zone changes, such as was the case with the "uk" TLD in 2014.

Subordinate and Superordinate: These terms are introduced in [RFC3731] for use in the registration model, but not defined there. Instead, they are given in examples. "For example, domain name 'example.com' has a superordinate relationship to host name ns1.example.com'." "For example, host ns1.example1.com is a

subordinate host of domain example1.com, but it is a not a subordinate host of domain example2.com." (Quoted from [RFC3731], Section 1.1.) These terms are strictly ways of referring to the relationship standing of two domains where one is a subdomain of the other.

10. General DNSSEC

Most DNSSEC terms are defined in [RFC4033], [RFC4034], [RFC4035], and [RFC5155]. The terms that have caused confusion in the DNS community are highlighted here.

DNSSEC-aware and DNSSEC-unaware: These two terms, which are used in some RFCs, have not been formally defined. However, Section 2 of [RFC4033] defines many types of resolvers and validators, including "non-validating security-aware stub resolver", "non-validating stub resolver", "security-aware name server", "security-aware recursive name server", "security-aware resolver", "security-aware stub resolver", and "security-oblivious 'anything'". (Note that the term "validating resolver", which is used in some places in DNSSEC-related documents, is also not defined in those RFCs, but is defined below.)

Signed zone: "A zone whose RRsets are signed and that contains properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records." (Quoted from [RFC4033], Section 2.) It has been noted in other contexts that the zone itself is not really signed, but all the relevant RRsets in the zone are signed. Nevertheless, if a zone that should be signed contains any RRsets that are not signed (or opted out), those RRsets will be treated as bogus, so the whole zone needs to be handled in some way.

It should also be noted that, since the publication of [RFC6840], NSEC records are no longer required for signed zones: a signed zone might include NSEC3 records instead. [RFC7129] provides additional background commentary and some context for the NSEC and NSEC3 mechanisms used by DNSSEC to provide authenticated denial-of-existence responses. NSEC and NSEC3 are described below.

Unsigned zone: Section 2 of [RFC4033] defines this as "a zone that is not signed". Section 2 of [RFC4035] defines this as "A zone that does not include these records [properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records] according to the rules in this section". There is an important note at the end of Section 5.2 of [RFC4035] that defines an additional situation in which a zone is considered unsigned: "If the resolver does not support any of the algorithms

listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned."

NSEC: "The NSEC record allows a security-aware resolver to authenticate a negative reply for either name or type non-existence with the same mechanisms used to authenticate other DNS replies." (Quoted from [RFC4033], Section 3.2.) In short, an NSEC record provides authenticated denial of existence.

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

NSEC3: Like the NSEC record, the NSEC3 record also provides authenticated denial of existence; however, NSEC3 records mitigate against zone enumeration and support Opt-Out. NSEC3 resource records require associated NSEC3PARAM resource records. NSEC3 and NSEC3PARAM resource records are defined in [RFC5155].

Note that [RFC6840] says that [RFC5155] "is now considered part of the DNS Security Document Family as described by Section 10 of [RFC4033]". This means that some of the definitions from earlier RFCs that only talk about NSEC records should probably be considered to be talking about both NSEC and NSEC3.

Opt-out: "The Opt-Out Flag indicates whether this NSEC3 RR may cover unsigned delegations." (Quoted from [RFC5155], Section 3.1.2.1.) Opt-out tackles the high costs of securing a delegation to an insecure zone. When using Opt-Out, names that are an insecure delegation (and empty non-terminals that are only derived from insecure delegations) don't require an NSEC3 record or its corresponding RRSIG records. Opt-Out NSEC3 records are not able to prove or deny the existence of the insecure delegations. (Adapted from [RFC7129], Section 5.1)

Insecure delegation: "A signed name containing a delegation (NS RRset), but lacking a DS RRset, signifying a delegation to an unsigned subzone." (Quoted from [RFC4956], Section 2.)

Zone enumeration: "The practice of discovering the full content of a zone via successive queries." (Quoted from [RFC5155], Section 1.3.) This is also sometimes called "zone walking". Zone enumeration is different from zone content guessing where the guesser uses a large dictionary of possible labels and sends

successive queries for them, or matches the contents of NSEC3 records against such a dictionary.

Validation: Validation, in the context of DNSSEC, refers to one of the following:

* Checking the validity of DNSSEC signatures

* Checking the validity of DNS responses, such as those including authenticated denial of existence

* Building an authentication chain from a trust anchor to a DNS response or individual DNS RRsets in a response

The first two definitions above consider only the validity of individual DNSSEC components such as the RRSIG validity or NSEC proof validity. The third definition considers the components of the entire DNSSEC authentication chain, and thus requires "configured knowledge of at least one authenticated DNSKEY or DS RR" (as described in [RFC4035], Section 5).

[RFC4033], Section 2, says that a "Validating Security-Aware Stub Resolver... performs signature validation" and uses a trust anchor "as a starting point for building the authentication chain to a signed DNS response", and thus uses the first and third definitions above. The process of validating an RRSIG resource record is described in [RFC4035], Section 5.3.

[RFC5155] refers to validating responses throughout the document, in the context of hashed authenticated denial of existence; this uses the second definition above.

The term "authentication" is used interchangeably with "validation", in the sense of the third definition above. [RFC4033], Section 2, describes the chain linking trust anchor to DNS data as the "authentication chain". A response is considered to be authentic if "all RRsets in the Answer and Authority sections of the response [are considered] to be authentic" (Quoted from [RFC4035]). DNS data or responses deemed to be authentic or validated have a security status of "secure" ([RFC4035], Section 4.3; [RFC4033], Section 5). "Authenticating both DNS keys and data is a matter of local policy, which may extend or even override the [DNSSEC] protocol extensions" (Quoted from [RFC4033], Section 3.1).

The term "verification", when used, is usually synonym for "validation".

Validating resolver: A security-aware recursive name server, security-aware resolver, or security-aware stub resolver that is applying at least one of the definitions of validation (above), as appropriate to the resolution context. For the same reason that the generic term "resolver" is sometimes ambiguous and needs to be evaluated in context (see Section 6), "validating resolver" is a context-sensitive term.

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

Zone signing key (ZSK): "DNSSEC keys that can be used to sign all the RRsets in a zone that require signatures, other than the apex DNSKEY RRset." (Quoted from [RFC6781], Section 3.1) Also note that a ZSK is sometimes used to sign the apex DNSKEY RRset.

Combined signing key (CSK): "In cases where the differentiation between the KSK and ZSK is not made, i.e., where keys have the role of both KSK and ZSK, we talk about a Single-Type Signing Scheme." (Quoted from [RFC6781], Section 3.1) This is sometimes called a "combined signing key" or CSK. It is operational practice, not protocol, that determines whether a particular key is a ZSK, a KSK, or a CSK.

Secure Entry Point (SEP): A flag in the DNSKEY RDATA that "can be used to distinguish between keys that are intended to be used as the secure entry point into the zone when building chains of trust, i.e., they are (to be) pointed to by parental DS RRs or configured as a trust anchor. Therefore, it is suggested that the SEP flag be set on keys that are used as KSKs and not on keys that are used as ZSKs, while in those cases where a distinction between a KSK and ZSK is not made (i.e., for a Single-Type Signing Scheme), it is suggested that the SEP flag be set on all keys." (Quoted from [RFC6781], Section 3.2.3.) Note that the SEP flag is only a hint, and its presence or absence may not be used to disqualify a given DNSKEY RR from use as a KSK or ZSK during validation.

The original definition of SEPs was in [RFC3757]. That definition clearly indicated that the SEP was a key, not just a bit in the key. The abstract of [RFC3757] says: "With the Delegation Signer (DS) resource record (RR), the concept of a public key acting as a secure entry point (SEP) has been introduced. During exchanges of public keys with the parent there is a need to differentiate SEP keys from other public keys in the Domain Name System KEY (DNSKEY) resource record set. A flag bit in the DNSKEY RR is defined to indicate that DNSKEY is to be used as a SEP." That definition of

the SEP as a key was made obsolete by [RFC4034], and the definition from [RFC6781] is consistent with [RFC4034].

Trust anchor: "A configured DNSKEY RR or DS RR hash of a DNSKEY RR. A validating security-aware resolver uses this public key or hash as a starting point for building the authentication chain to a signed DNS response. In general, a validating resolver will have to obtain the initial values of its trust anchors via some secure or trusted means outside the DNS protocol." (Quoted from [RFC4033], Section 2)

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

DNSSEC Practice Statement (DPS): "A practices disclosure document that may support and be a supplemental document to the DNSSEC Policy (if such exists), and it states how the management of a given zone implements procedures and controls at a high level." (Quoted from [RFC6841], Section 2)

Hardware security module (HSM): A specialized piece of hardware that is used to create keys for signatures and to sign messages without ever disclosing the private key. In DNSSEC, HSMs are often used to hold the private keys for KSKs and ZSKs and to create the signatures used in RRSIG records at periodic intervals.

Signing software: Authoritative DNS servers that support DNSSEC often contain software that facilitates the creation and maintenance of DNSSEC signatures in zones. There is also stand-alone software that can be used to sign a zone regardless of whether the authoritative server itself supports signing. Sometimes signing software can support particular HSMs as part of the signing process.

11. DNSSEC States

A validating resolver can determine that a response is in one of four states: secure, insecure, bogus, or indeterminate. These states are defined in [RFC4033] and [RFC4035], although the two definitions differ a bit. This document makes no effort to reconcile the two definitions, and takes no position as to whether they need to be reconciled.

Section 5 of [RFC4033] says:

A validating resolver can determine the following 4 states:

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

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

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

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

Section 4.3 of [RFC4035] says:

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

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

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

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

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

12. Security Considerations

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

13. IANA Considerations

None.

14. References

14.1. Normative References

[IANA_RootFiles]
Internet Assigned Numbers Authority, "IANA Root Files", 2016, <http://www.iana.org/domains/root/files>.

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

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

[RFC1912] Barr, D., "Common DNS Operational and Configuration Errors", RFC 1912, DOI 10.17487/RFC1912, February 1996, <https://www.rfc-editor.org/info/rfc1912>.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[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>.

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

[RFC2182] Elz, R., Bush, R., Bradner, S., and M. Patton, "Selection and Operation of Secondary DNS Servers", BCP 16, RFC 2182, DOI 10.17487/RFC2182, July 1997, <https://www.rfc-editor.org/info/rfc2182>.

[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>.

[RFC3731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", RFC 3731, DOI 10.17487/RFC3731, March 2004, <https://www.rfc-editor.org/info/rfc3731>.

[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>.

[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>.

[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>.

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

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5855] Abley, J. and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", BCP 155, RFC 5855, DOI 10.17487/RFC5855, May 2010, <https://www.rfc-editor.org/info/rfc5855>.

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

[RFC6561] Livingood, J., Mody, N., and M. O'Reirdan, "Recommendations for the Remediation of Bots in ISP Networks", RFC 6561, DOI 10.17487/RFC6561, March 2012, <https://www.rfc-editor.org/info/rfc6561>.

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

[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>.

[RFC6841] Ljunggren, F., Eklund Lowinder, AM., and T. Okubo, "A Framework for DNSSEC Policies and DNSSEC Practice Statements", RFC 6841, DOI 10.17487/RFC6841, January 2013, <https://www.rfc-editor.org/info/rfc6841>.

[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>.

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

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

[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>.

14.2. Informative References

[I-D.ietf-doh-dns-over-https]
Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", draft-ietf-doh-dns-over-https-14 (work in progress), August 2018.

[IANA_Resource_Registry]
Internet Assigned Numbers Authority, "Resource Record (RR) TYPEs", 2017, <http://www.iana.org/assignments/dns-parameters/>.

[RFC0819] Su, Z. and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC 819, DOI 10.17487/RFC0819, August 1982, <https://www.rfc-editor.org/info/rfc819>.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

[RFC1713] Romao, A., "Tools for DNS debugging", FYI 27, RFC 1713, DOI 10.17487/RFC1713, November 1994, <https://www.rfc-editor.org/info/rfc1713>.

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

[RFC2133] Gilligan, R., Thomson, S., Bound, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 2133, DOI 10.17487/RFC2133, April 1997, <https://www.rfc-editor.org/info/rfc2133>.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <https://www.rfc-editor.org/info/rfc2775>.

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[RFC3425] Lawrence, D., "Obsoleting IQUERY", RFC 3425, DOI 10.17487/RFC3425, November 2002, <https://www.rfc-editor.org/info/rfc3425>.

[RFC3757] Kolkman, O., Schlyter, J., and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag", RFC 3757, DOI 10.17487/RFC3757, April 2004, <https://www.rfc-editor.org/info/rfc3757>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, DOI 10.17487/RFC4641, September 2006, <https://www.rfc-editor.org/info/rfc4641>.

[RFC4697] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", BCP 123, RFC 4697, DOI 10.17487/RFC4697, October 2006, <https://www.rfc-editor.org/info/rfc4697>.

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

[RFC4956] Arends, R., Kosters, M., and D. Blacka, "DNS Security (DNSSEC) Opt-In", RFC 4956, DOI 10.17487/RFC4956, July 2007, <https://www.rfc-editor.org/info/rfc4956>.

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

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[RFC5893] Alvestrand, H., Ed. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.

[RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, DOI 10.17487/RFC6055, February 2011, <https://www.rfc-editor.org/info/rfc6055>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <https://www.rfc-editor.org/info/rfc6303>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <https://www.rfc-editor.org/info/rfc6365>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <https://www.rfc-editor.org/info/rfc6672>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <https://www.rfc-editor.org/info/rfc7129>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

[RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin, "Inventory and Analysis of WHOIS Registration Objects", RFC 7485, DOI 10.17487/RFC7485, March 2015, <https://www.rfc-editor.org/info/rfc7485>.

[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>.

[RFC8109] Koch, P., Larson, M., and P. Hoffman, "Initializing a DNS Resolver with Priming Queries", BCP 209, RFC 8109, DOI 10.17487/RFC8109, March 2017, <https://www.rfc-editor.org/info/rfc8109>.

[RSSAC026] Root Server System Advisory Committee (RSSAC), "RSSAC Lexicon", 2017, <https://www.icann.org/en/system/files/files/rssac-026-14mar17-en.pdf>.

Appendix A. Definitions Updated by this Document

The following definitions from RFCs are updated by this document:

- Forwarder in [RFC2308]
- QNAME in [RFC2308]
- Secure Entry Point (SEP) in [RFC3757]; note, however, that this RFC is already obsolete

Appendix B. Definitions First Defined in this Document

The following definitions are first defined in this document:

- "Alias" in Section 2
- "Apex" in Section 7
- "arpa" in Section 7
- "Bailiwick" in Section 7
- "Class independent" in Section 5
- "Delegation-centric zone" in Section 7
- "Delegation" in Section 7
- "DNS operator" in Section 9
- "DNSSEC-aware" in Section 10
- "DNSSEC-unaware" in Section 10
- "Forwarding" in Section 6
- "Full resolver" in Section 6
- "Fully qualified domain name" in Section 2
- "Global DNS" in Section 2
- "Hardware Security Module (HSM)" in Section 10
- "Host name" in Section 2
- "IDN" in Section 2
- "In-bailiwick" in Section 7
- "Iterative resolution" in Section 6
- "Label" in Section 2
- "Locally served DNS zone" in Section 2
- "Naming system" in Section 2

- "Negative response" in Section 3
- "Non-recursive query" in Section 6
- "Open resolver" in Section 6
- "Out-of-bailiwick" in Section 7
- "Passive DNS" in Section 6
- "Policy-implementing resolver" in Section 6
- "Presentation format" in Section 5
- "Priming" in Section 6
- "Private DNS" in Section 2
- "Recursive resolver" in Section 6
- "Referrals" in Section 4
- "Registrant" in Section 9
- "Registrar" in Section 9
- "Registry" in Section 9
- "Root zone" in Section 7
- "Secure Entry Point (SEP)" in Section 10
- "Signing software" in Section 10
- "Split DNS" in Section 6
- "Stub resolver" in Section 6
- "Subordinate" in Section 8
- "Superordinate" in Section 8
- "TLD" in Section 2
- "Validating resolver" in Section 10
- "Validation" in Section 10

- "View" in Section 6
- "Zone transfer" in Section 6

Index

Acknowledgements

The following is the Acknowledgements for RFC 7719.

The authors gratefully acknowledge all of the authors of DNS-related RFCs that proceed this one. Comments from Tony Finch, Stephane Bortzmeyer, Niall O'Reilly, Colm MacCarthaigh, Ray Bellis, John Kristoff, Robert Edmonds, Paul Wouters, Shumon Huque, Paul Ebersman, David Lawrence, Matthijs Mekking, Casey Deccio, Bob Harold, Ed Lewis, John Klensin, David Black, and many others in the DNSOP Working Group helped shape RFC 7719.

Most of the major changes between RFC 7719 and this document came from active discussion on the DNSOP WG. Specific people who contributed material to this document include: Bob Harold, Dick Franks, Evan Hunt, John Dickinson, Mark Andrews, Martin Hoffmann, Paul Vixie, Peter Koch, Duane Wessels, Allison Mankin, Giovane Moura, Roni Even, Dan Romascanu, and Vladmir Cunat.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Andrew Sullivan

Email: ajs@anvilwalrusden.com

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

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

DNSOP
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

D. Migault
Ericsson
D. York
Internet Society
E. Lewis
ICANN

October 30, 2017

DNSSEC Validators Requirements
draft-mglt-dnsop-dnssec-validator-requirements-06

Abstract

DNSSEC provides data integrity and source authentication to a basic DNS RReet. Given a RRset, a public key and a signature, a DNSSEC validator checks the signature, time constraints, and other, local, policies. In case of mismatch the RRSet is considered illegitimate and is rejected.

Accuracy in DNSSEC validation, that is, avoiding false positives and catching true negatives, requires that both the signing process and validation process adhere to the protocol, which begins with external configuration parameters. This document describes requirements for a validator to be able to perform accurate validation.

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 3, 2018.

Copyright Notice

Table of Contents

1. Requirements notation

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

2. Introduction

DNSSEC validation [RFC4033], [RFC4034] and [RFC4035] has two core concepts. One is the matching of a RRSIG resource record's contents to a RRset, making use of a DNSKEY resource record (named in the

RRSIG record). Two is the placing of trust in the DNSKEY resource record.

Evaluation based on a RRSIG record involves a few steps. Most visible is a cryptographic operation, matching the digital signature in the RRSIG with the specified public key in the named DNSKEY record and a properly prepared DNS RRset. This is meant to demonstrate that the RRset came from an entity with the private component of the key (source authenticity).

Not to be forgotten are the other matches to perform. To address the threat of reply attacks, wall-clock (absolute) time is checked. To address the authority of the source, the named DNSKEY record is checked for appropriateness (i.e., owned by the same zone is the default policy).

The RRSIG record also contains other information intended to help the validator perform its work, in some cases "sane value" checks are performed. For instance, the original TTL (needed to prepare the RR set for validation) ought to be equal to or higher than the received TTL.

Requirements related to validation exist in [RFC4033], [RFC4034] and [RFC4035]. However, the specification of the validation is not sufficient to enable a wide deployment of DNSSEC validators. In fact, there are a number of situations where the necessary condition are not met by the DNSSEC validator to perform DNSSEC validation. When such conditions are not met, the DNSSEC validation may qualify improperly a RRset as invalid. This document is focused on the necessary mechanisms that DNSSEC validators should implement in order to make DNSSEC validation output accurate. The mechanisms described in this document include, provisioning mechanisms as well as monitoring and management mechanisms that enables an administrator to validate the validity of the DNSSEC validation output.

3. Terminology

This document uses the following terminology:

DNSSEC validator: the entity that performs DNSSEC resolution and performs signature validation.

Accurate validation: validation that avoids false positives and catches true negatives. (not sure if this is needed, but seems appropriate)

Trust Anchor Data Store:

4. DNSSEC Validator Description

This is a conceptual block diagram of the elements involved with DNSSEC validation. This is not meant to be an architecture for code, this is meant to be a framework for discussion and explanation.

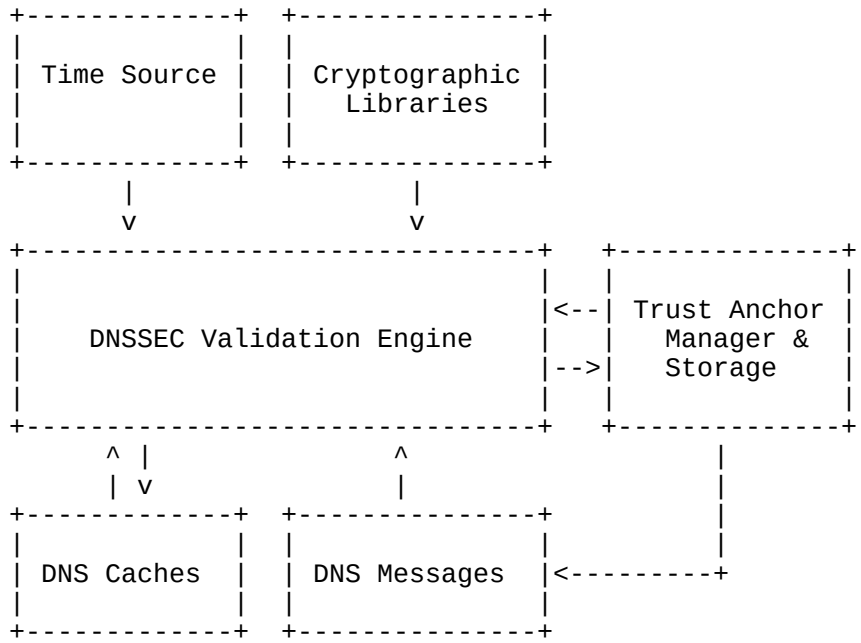


Figure 1: DNSSEC Validator Description

Time Source : Wall clock time Cryptograhic Libraries: Code performing mathematical functions.

DNS Message : Receiver of DNS responses DNS Caches: Positive and negative caches.

Trust Anchor Manager : database of trust anchors, manages trust DNSSEC Validation Engine: follows local policy to approve data.

a. Time Source -> DNSSEC Validation Engine Current time upon request, in appropriate time zone setting

b. Cryptographic Libraries-> DNSSEC Validation Engine Supplies code to perform math, the engine determines the DNSSEC Security Algorithms supported

c. DNS Caches <- DNSSEC Validation Engine Enter the results of a validation (positive data, negative failures)

d. DNS Caches -> DNSSEC Validation Engine Stored keys, etc., used in building a chain of trust

e. DNS Messages -> DNSSEC Validation Engine DNS Responses needed validation

f. Trust Anchor Management & Storage -> DNSSEC Validation Engine Supplies trust anchor information when needed.

g. Trust Anchor Management & Storage <- DNSSEC Validation Engine Information to update the trust anchor store, resulting from automated update requests.

h. Trust Anchor Management & Storage -> DNS Messages Requests made to manage trust anchors.

i. Not shown - Name Server Process Management interfaces to elements, handling of Checking Disabled request, responses, as well as all API requests made of the name server.

5. Time derivation and absence of Real Time Clock

With M2M communication some devices are not expecting to embed Real Time Clock (Raspberry Pi is one example of such devices). When these devices are re-plugged the initial time is set to January 1 1970. Other devices that have clocks that may suffer from time derivation. All these devices cannot rely on their time estimation to perform DNSSEC validation.

REQ1: A DNSSEC validator MUST be provided means to update the time without relying on DNSSEC.

Note that updating time in order to be able to perform DNSSEC validation may easily come with a chicken-and-egg problem when the NTP server is designated by its FQDN. The update mechanisms must consider the DNSSEC validator may not able to validate the DNSSEC queries. In other words, the mechanisms may have to update the time over an unsecure DNSSEC resolution.

6. Trust Anchor

6.1. Trust Anchor Bootstrapping

A validator needs to have trust anchors or it will never be able to construct a chain of trust. Trust anchors are defined by DNSSEC to be keys that are inherently trusted, configured by authorized parties, in the validator. The configuration can be via an automated process, such as Automated Updates of DNSSEC Trust Anchors [RFC5011], [I-D.ietf-dnsop-rfc5011-security-considerations], or via manual process.

An implementation of a validator needs to allow an operator to choose any automated process supported by the validator. (No requirements are stated about what processes to support, only one is standardized to date.) An implementation needs to also afford the operator the ability to override or manage via a purely manual process, the storage of managed keys. This includes adding, deleting, changing and inspecting.

Beyond the scope of these requirements are the decision processes of authorized parties in placing trust in keys.

REQ2: A DNSSEC validator MUST check the validity of its Trust Anchors. When a Trust Anchor cannot be verified, the DNSSEC validator MUST send a warning and SHOULD NOT start validating traffic without manual validation.

REQ3: A DNSSEC validator SHOULD be able to retrieve a Trust Anchor with bootstrapping mechanism. Such mechanism' security MUST NOT be based on DNSSEC, but could instead include downloading a XML file from a trusted URL, or a PKIX certificate.

Although some bootstrapping mechanisms to securely retrieve publish [RFC7958] and retrieve [UNBOUND-ANCHOR] the Root Zone Trust Anchor have been defined, it is believed these mechanisms should be extended to other KSKs or Trust Anchors. In fact it is not always possible to build a trusted delegation between the Root Zone and any sub zone. This may happen for example if one of the upper zones does not handle the secure delegation or improperly implement it. A DS RRset may not be properly filled or its associated signature cannot be validated. As the chain of trust between a zone and the root zone may not be validated, the DNSSEC validation for the zone requires a Trust Anchor. Such DNS(SEC) resolutions may be critical for infrastructure management. A company "Example" may, for example, address all its devices under the domain example.com and may not want disruption to happen if the .com delegation cannot be validated for any reason. Such companies may provision there DNSSEC validator with the Trust Anchor KSK for the zone example.com in addition to the regular DNSSEC delegation. Similarly some some domains may present different views

such as a "private" view and a "public view". These zones may have some different content, and may use a different KSK for each view.

6.2. Trust Anchor Data Store

When DNSSEC validator are running and a Trust Anchor KSK roll over is ongoing, a network administrator or any trust party may be willing to check whether the new published keys are being stored in a Trust Anchor Data Store with an appropriated status. Such inspection aims at detecting an non successful Trust Anchor roll over before traffic is being rejected. When a new Trust Anchor has not been considered by the DNSSEC validator, a trusted party may be able to provision the DNSSEC validator with the new Trust Anchor, and eventually may remove the revoked Trust Anchor.

While using a Trust Anchor that has been removed results in the DNSSEC validator rejecting multiple legitimate responses, the consequences associated to accepting a rogue Trust Anchor as a legitimate Trust Anchor are even worst. Such attacks would result in an attacker taking control of the entire naming space behind the Trust Anchor. In the case of the Root Zone KSK, for example, almost all name space would be under the control of the attacker. In addition, to the name space, once the rogue Trust Anchor is configured, there is little hope the DNSSEC validator be re-configured with the legitimate Trust Anchor without manual intervention. As a result, it is crucial to cautiously handle operations related to the Trust Anchor provisioning. Means must be provided so network administrator can clearly diagnose the reason a Trust Anchor is not valid to avoid accepting a rogue Trust Anchor inadvertently.

DNSSEC may also be used in some private environment. Corporate networks and home networks, for example, may want to take advantage of DNSSEC for a local scope network. Typically, a corporate network may use a local scope Trust Anchor to validate DNS RRsets provided by authoritative DNSSEC server in the corporate network. This use case is also known as the "split-view" use case. These RRsets within the corporate network may differ from those hosted on the public DNS infrastructure. Note that using different Trust Anchor for a given zone may expose a zone to signature invalidation. This is especially the case for DNSSEC validators that are expected to flip-flop between local and public scope. How validators have to handle the various provisioned Trust Anchors is out of scope of the document.

Home network may use DNSSEC with TLDs or associated domain names that are of local scope and not even registered in the public DNS infrastructure. This requires the ability to manage the Trust Anchor as well.

The necessity to interact with the Trust Anchors lead to the following requirements:

REQ4: A DNSSEC validator MUST store its Trust Anchors in a dedicated Trust Anchor Data Store. Such database MUST store informations associated to each Trust Anchor status as well as the time the status has been noticed by the DNSSEC validator. Such database MUST be resilient to DNSSEC validator reboot.

REQ5: Trust Anchor states SHOULD at least consider those described in [RFC5011] (Start, AddPend, Valid, Missing, Revoked, Removed). Additional states SHOULD also be able to indicate additional motivations for revoking the Trust Anchor such as a Trust Anchor known to be corrupted, a Trust anchor miss published, or part of a regular roll over procedure.

REQ6: A DNSSEC validator MUST provide access to the Trust Anchor Data Sase to authorized user only. Access control is expected to be based on a least privileged principles.

REQ7: A trusted party MUST be able to add, remove a Trust Anchor in the Trust Anchor Data Store.

6.3. Interactions with the cached RRsets

In addition when a Trust Anchor is revoked, the DNSSEC validator may behave differently if the revocation is motivated by a regular roll over operation or instead by revoking a Trust Anchor that is known as being corrupted. In the case the roll over procedure, is motivated by revoking a Trust Anchor that is known to be corrupted, the DNSSEC validator may be willing to flush all RRsets that depends on the Trust Anchor.

REQ8: A DNSSEC validator MUST be able to flush the cached RRsets that rely on a Trust Anchor.

7. ZSK / KSK

7.1. KSK/ZSK Data Store

A number of reasons may result in inconsistencies between the RRsets stored in the cache and those published by the authoritative server.

An emergency KSK / ZSK rollover may result in a new KSK / ZSK with associated new RRSIG published in the authoritative zone, while DNSSEC validator may still cache the old value of the ZSK / KSK. For a RRset not cached, the DNSSEC validator performs a DNSSEC query to the authoritative server that returns the RRset signed with the new

KSK / ZSK. The DNSSEC validator may not be able to retrieve the new KSK / ZSK while being unable to validate the signature with the old KSK / ZSK. This either result in a bogus resolution or in an invalid signature check. Note that by comparing the Key Tag Fields, the DNSSEC validator is able to notice the new KSK / ZSK used for signing differs from the one used to generate the received generated signature. However, the DNSSEC validator is not expectected to retrieve the new ZSK / KSK, as such behavior could be used by an attacker. Intsead, ZSK / ZSK key roll ove rprocedure are expected to avoid such inconsistencies.

Similarly, a KSK / ZSK roll over may be performed normally, that is as described in [RFC6781] and [RFC7583]. While the KSK / ZSK roll over is performed, there is no obligation to flush the RRsets in the cache that have been associated with the old key. In fact, these RRset may still be considered as trusted and be removed from the cache as their TTL timeout. With very long TTL, these RRset may remain in the cache while the ZSK / KSK with a shorter TTL is no longer published nor in the cache. In such situations, the purpose of the KSK / ZSK is to validate the data is considered trusted at the time it enters the cache, and such trust may remain after the KSK / ZSK is being rolled over. Note also that even though the data may not be associated to the KSK / ZSK that has been used to valiadte the data, the link between the KSK / ZSK and teh data is still stored in teh cache using the RRSIG. Note also that inconsistencies between the ZSK / KSK stored in the cache and those published on the authoritative server, may lead to inconsistencies to downstream DNSSEC validators that realy on multiple cache over time. Typically, a request for the KSK / ZSK may have been provided by a cache that is storing the new published value, while the data and associated sigature may be associated to the old KSK / ZSK.

KSK and ZSK are associated with configuration parameters, and as such are expected to be stored only in the cache. As a result, flushing their value from the cache could constitute a way forward to refresh them. On the other hand, their respective function is also to prevent illegitimate RRsets to be validated and so more understanding is need before taking any action associated to the KSK or ZSK. More specifically, the network administrator SHOULD be provided the appropriated information required to distinguish a misconfiguration from an attack.

The following requirements are thus considered for the KSK / ZSK.

REQ9: A DNSSEC validator MUST store its KSK/ZSK in a dedicated KSK/ ZSK Data Base. Such database MUST store informations associated to each KSK/ZSK status as well as the time the status has been noticed by the DNSSEC validator. Such

database MUST NOT be resilient to DNSSEC validator reboot, that is the information stored in the Data Base MUST NOT be used to populate the cache, while it MAY be used as second factor verification, or audit for example.

REQ10: KSK/ZSK status and informaton SHOULD be monitored continuously and associated with their respective state as well as verified time. These states and time SHOULD be resilient to reboot.

REQ11: KSK/ZSK states SHOULD at least consider those described in section 3.1 of [RFC7583] (Generated, Published, Ready, Active, Retired, Dead, Removed, Revoked). Additional states SHOULD also be able to indicate additional motivations for revoking the KSK/ZSK such as a KSK/ZSK known to be corrupted, a KSK/ZSK miss published, or part of a regular roll over procedure.

REQ12: A DNSSEC validator MUST provide access to the KSK/ZSK data base to authorized user only. Access control is expected to be based on a least privileged principles.

REQ13: A trusted party MUST be able to add, remove a Trust Anchor in the KSK/ZSK Database.

Similarly to its counter part the TA Data Store, the KSK/ZSK Data Store is expected to be resilient to reboot. However the motivation for having the KSK/ZSK Data Store resilient to reboot differs from those for making the TA Data Store resilient to reboot. TA Data Store needs to be resilient as the Trust Anchors are necessary to perform the DNSSEC validation. KSK/ZSK are not expected to be locally stored, but instead are expected to be resolved, validated by the TA and stored in the cache. The reason for making the KSK/ZSK Data Store resilient to reboot is mostly to enable audit of the DNSSEC validator.

7.2. KSK/ZSK Data Store and Trust Anchor Data Store

A zone may have been badly signed, which means that the KSK or ZSK cannot validate the RRSIG associated to the RRsets. This may not be due to a key roll over, but to an incompatibility between the keys (KSK or ZSK) and the signatures.

When such situation occurs, there is only a choice between not validating the RRsets or invalidating their signature. This is a policy design that needs to be taken by the network administrator. In other ways, flushing the RRset are not expected to address this issue. Such KSK/ZSK are known as Negative Trust Anchors [RFC7646].

With Negative Trust Anchor, the zone for a given time will be known as "known insecure". The DNSSEC Validator is not expected to perform signature validation for this zone. It is expected that this information is associated to a Time To Live (TTL).

Note that, this information may be used as an attack vector to impersonate a zone, and must be provided in a trusted way, by a trusted party.

If a zone has been badly signed, the administrator of the authoritative DNS server may resign the zone with the same keys or proceed to an emergency key rollover. If the signature is performed with the same keys, the DNSSEC Validator may notice by itself that RRSIG can be validated. On the other hand if a key rollover is performed, the newly received RRSIG will carry a new key id. Upon receiving a new key id in the RRSIG, the DNSSEC Validator is expected to retrieve the new ZSK/KSK. If the RRSIG can be validated, the DNSSEC Validator is expected to remove the "known insecure" flag.

However, if the KSK/ZSK are rolled over and RRSIG cannot be validated, it remains hard for the DNSSEC validator to determine whether the RRSIG cannot be validated or that RRSIG are invalid. As a result:

REQ14: A trusted party MUST be able to indicate a DNSSEC validator that a KSK or a ZSK as Negative Trust Anchor. Such Trust Anchors MUST NOT be used for RRSIG validation and MUST be moved to the Trust Anchor Data Store, so the information become resilient to reboot.

REQ15: A trusted party MUST be able to indicate a DNSSEC validator that a KSK/ZSK is known "back to secure".

7.3. Interactions with cached RRsets

The key roll over procedure intends to ensure that the published RRsets can be validated with the KSK / ZSK stored in the various cache of the DNSSEC validators. As a consequence, the key roll over enables trusted data to be cached. However, the key roll over does not necessarily prevents that cached be always validated with the currenlty published key. In fact, a cached data may have been validated by the former key and remain in the cache while the former key has been rolled out. Such inconsistencies may be acceptable and correspond to the following trust model: the KSK / ZSK validate the cached data can be trusted at time T. There is no specific information that leads to considers that trust at time T is subject to doubts at current time, so the cached data remain trusted.

While such inconsistencies may have little impact on end host DNSSEC validators, it may be different for large resolving platforms with downstream DNSSEC validators, and a DNSSEC validator may be willing to maintain its cached data consistent with the published KSK / ZSK. A trusted third party may willing to remove all cached RRsets that have been validated by the KSK/ZSK upon some specific states (revoked, or Removed for example), of after some time after the state is noticed. In this later case, only the RRset whose TTL has not expired yet would be flushed.

On the other hand, when a KSK / ZSK is known to be corrupted, this state may affect the trust that has been established at time T. In such case, the DNSSEC validator may be willling to flush all cached data that has been validated by the currently known corrupted KSK / ZSK, including the KSK / ZSK itslef.

As a result, the following requirements are expected:

REQ16: A DNSSEC validator MUST be able to flush the cached KSK/ZSK.

REQ17: A DNSSEC validator MUST be able to flush the cached RRsets associated to a KSK/ZSK.

8. DS

The DS RRset is stored in the parent zone to build a chain of trust with the child zone. This DS RRset can be invalid because its RDATA (KSK) is not anymore used in the child zone or because the DS is badly signed and cannot be validated by the DNSSEC Validator.

In both cases the child zone is considered as bogus and the valid child zone’s KSK should become a Trust Anchor KSK. This requirements is fulfilled by the requirement to add a Trust Anchor in Section 6.

9. Cryptography Deprecation

As mentioned in [RFC8247] and [RFC8221] cryptography used one day is expected over the time to be replaced by new and more robust cryptographic mechanisms. In the case of DNSSEC signature protocols are likely to be updated over time. In order to anticipate the sunset of one of the signature scheme, a DNSSEC validator may willing to estimate the impact of deprecating one signature scheme.

Currently [RFC6975] provides the ability for a DNSSEC validator to announce an authoritative server the supported signature schemes. However, a DNSSEC validator is not able to determine other than by trying whether a signature scheme is supported by the authoritative server.

In order for a DNSSEC validator to safely deprecate one signature scheme the following requirement should be fulfilled.

REQ18: A DNSSEC validator SHOULD be able to request the signature scheme supported by an authoritative server.

10. Reporting

A DNSSEC validator receiving a DNS response cannot make the difference between receiving an non-secure response versus an attack. Dropping DNSSEC fields by a misconfigured middle boxes, such as DS, RRRSIG is considered as an attack.

A DNSSEC validator is expected to perform secure DNS resolution and as such protect its stub client. An invalid response may be the result of an attack or a misconfiguration, and the DNSSEC validator may play an important role in sharing this information.

REQ19: A DNSSEC validation SHOULD be able to report the unavailability of the DNSSEC service.

REQ20: A DNSSEC validator SHOULD be able to report a invalid DNSSEC validation.

11. IANA Considerations

There are no IANA consideration for this document.

12. Security Considerations

The requirements listed in this document aim at providing the DNSSEC validator appropriated information so DNSSEC validation can be performed. On the other hand, providing inappropriate information can lead to misconfiguring the DNSSEC validator, and thus disrupting the DNSSEC resolution service. As a result, enabling the setting of configuration parameters by a third party may open a wide surface of attacks.

As an appropriate time value is necessary to perform signature check (cf. Section 5), an attacker may provide rogue time value to prevent the DNSSEC validator to check signatures.

An attacker may also affect the resolution service by regularly asking the DNSSEC validator to flush the KSK/ZSK from its cache (cf. Section 7). All associated data will also be flushed. This generates additional DNSSEC resolution and additional validations, as RRSet that were cached require a DNSSEC resolution over the Internet.

This affects the resolution service by slowing down responses, and increases the load on the DNSSEC validator.

An attacker may ask the DNSSEC validator to consider a rogue KSK/ZSK (cf. Invalid DS in Section 8 or Private KSK in Section 6), thus hijacking the DNS zone. Similarly, (cf. Section 7) an attacker may inform the DNSSEC validator not to trust a given KSK in order to prevent DNSSEC validation to be performed.

An attacker (cf. Section 7) can advertise a "known insecure" KSK or ZSK is "back to secure" to prevent signature check to be performed correctly.

As a result, information considered by the DNSSEC validator should be from a trusted party. This trust party should have been authenticated, and the channel used to exchange the information should also be protected and authenticated.

13. Acknowledgment

The need to address DNSSEC issues on the resolver side started in the Home Networks mailing list and during the IETF87 in Berlin. Among others, people involved in the discussion were Ted Lemon, Ralph Weber, Normen Kowalewski, and Mikael Abrahamsson. People involved in the email discussion initiated by Jim Gettys were, with among others, Paul Wouters, Joe Abley and Michael Richardson.

The current document has been initiated after a discussion with Paul Wouter and Evan Hunt.

14. References

14.1. 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>.

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

[RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC)", RFC 6975, DOI 10.17487/RFC6975, July 2013, <https://www.rfc-editor.org/info/rfc6975>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/info/rfc8221>.

[RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <https://www.rfc-editor.org/info/rfc8247>.

14.2. Informational References

[I-D.ietf-dnsop-rfc5011-security-considerations] Hardaker, W. and W. Kumari, "Security Considerations for RFC5011 Publishers", draft-ietf-dnsop-rfc5011-security-considerations-07 (work in progress), October 2017.

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

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

[RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J., and R. Weber, "Definition and Use of DNSSEC Negative Trust Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015, <https://www.rfc-editor.org/info/rfc7646>.

[RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <https://www.rfc-editor.org/info/rfc7958>.

[UNBOUND-ANCHOR] "unbound-anchor.c File Reference", <https://www.unbound.net/documentation/doxygen/unbound-anchor_8c.html#details>.

Authors' Addresses

Daniel Migault
Ericsson

Email: daniel.migault@ericsson.com

Dan York
Internet Society

Email: york@isoc.org

Edward Lewis
ICANN

Email: edward.lewis@icann.org

dnsop
Internet-Draft
Intended status: Informational
Expires: June 1, 2019

D. Migault
Ericsson
E. Lewis
ICANN
D. York
ISOC

November 28, 2018

DNSSEC Validator Requirements
draft-mglt-dnsop-dnssec-validator-requirements-07

Abstract

The DNS Security Extensions define a process for validating received data and assert them authentic and complete as opposed to forged.

This document describes what is needed in implementations to make the validation process manageable Considerations for accurate time as well as management of the trust anchor store.

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 June 1, 2019.

Table of Contents

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Introduction

The act of DNSSEC validation [RFC4033][RFC4035] can be broken into two part:

- Signature Validation: which consists in checking the cryptographic signature of a Resource Record Set (RRset). The signature validation involves among other parameters a DNSKEY Resource Record (RR) and RRSIG RR and the RRset itself. The signature

validation process results in assertion that the owner of the private part of the public key contained in the DNSKEY RR has effectively published the RRset. The binding between the private key and the RRset is provided by the trust in the cryptographic algorithm.

- Trust: Signature Validation results in asserting a RRset is accurately validated if there is sufficient trust that the owner of the private key associated to the DNSKEY RR is the owner of the RRset - i.e. that is to say is the legitimate owner. Such trust is provided by a Trust Anchor (TA), and the chain of trust established between the TA and the DNSKEY RR. The chain of trust is obtained by recursively validating the DNSKEY RRs. As a result, such trust results from the trust placed in the TA as well as the delegation mechanism provided by DNSSEC and the Signature Validation. As TAs need to be managed over time, the trust also concerns the management procedure of the TA. This is the main concern of this document.

Once accurately validated the RRset is assumed to be accurately validated and trusted trusted during the time indicated by its TTL.

A threat associated to the Signature Validation could consist in a RRSet maliciously forged to be validated by a trusted DNSKEY RR. Such threat mostly rely on the use of weak cryptography by the authoritative server, and the DNSSEC validator has little means to prevent such threats.

The document considers instead the threat associated to the establishment of the trust where a DNSKEY RR is maliciously established. This may be through a weakness in the authentication of changes to the zone administration database, allowing a malicious key to be added and then signed according to the DNSSEC process. Once this is discovered to have happened, other data validated via such a key should be called into question.

This document is focused on the necessary mechanisms that DNSSEC validators should implement in order to implement sufficient Trust that makes DNSSEC validation output accurate. The mechanisms described in this document include, provisioning mechanisms as well as monitoring and management mechanisms that enables an administrator to validate the validity of the DNSSEC validation output.

The mechanisms provided are designed in accordance of the DNSSEC trust model as to meet the current operations of DNSSEC. Such trust model is briefly recapped in Section 4 so operators understand the limits and motivations for such mechanisms.

3. Terminology

This document uses the following terminology:

DNSSEC validator the entity that performs DNSSEC resolution and performs signature validation.

Accurate validation validation that avoids false positives and catches true negatives.

Trust Anchor Data Store a module (of code) implementing functions related to the trust anchors used by the validator. This is essentially a database allowing access, monitoring of, and changes to trust anchors.

4. DNSSEC Validator Description

This is a conceptual block diagram of the elements involved with DNSSEC validation. This is not meant to be an architecture for code, this is meant to be a framework for discussion and explanation.

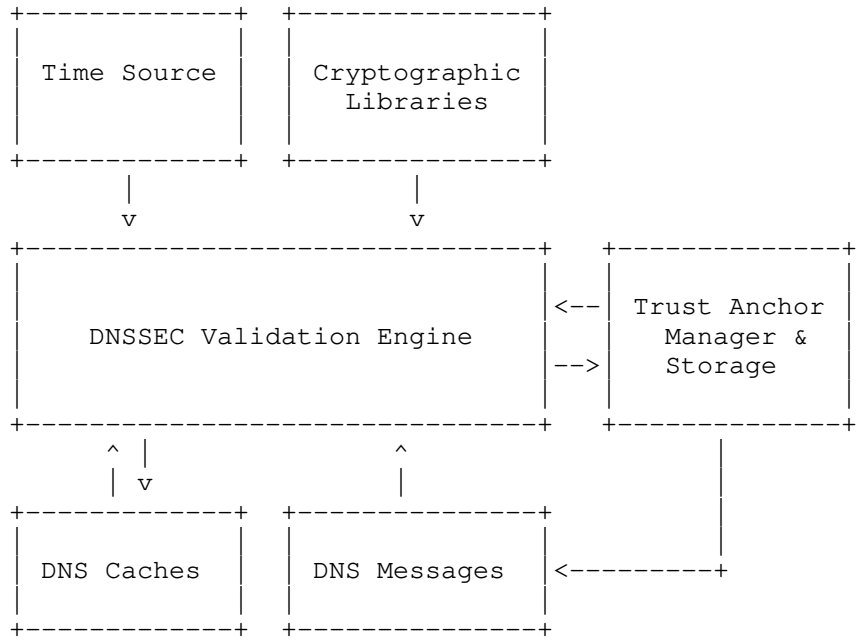


Figure 1: DNSSEC Validator Description

Time Source The wall clock time provides the DNSSEC Validation Engine the current time. Time is among other used to validate the

RRSIG Signature and Inception Fields to provide some protection against replay attacks.

Cryptograhic Libraries The code performing mathematical functions provides the DNSSEC Validation Engine the ability to check the Signature Field that contains the cryptographic signature covering the RRSIG RDATA.

DNS Message DNS responses are used to carry the information from the DNS system. The receiver of the DNS message can be any kind of application including DNS-related application such as in the case of automated Trust Anchor update performed by the Trust Anchor Manager & Storage. The DNSSEC Validator Engine accurately validates the DNS responses before caching them in the DNS Cache and forwarding them to the DNS receiver. In case of validation failure, an error is returned and the information may be negatively cached.

DNS Caches Include positive and negative caches. The DNSSEC Validation Engine fills DNS Caches with the results of a validation (positive data, negative failures). The DNSSEC trust model considers that once a RRset has been accurately validated by the DNSSEC Validator Engine, the RRset is considered trusted (or untrusted) for its associated TTL. DNS Caches contain RRsets that may contain information requested by the application (RRset of type AAAA for example) as well as RRset necessary to accurately validate the RRsets (RRsets of type DNSKET or RRSIG for example). It also worth noticing that RRset validated with DNSSEC or RRset that are not validated with DNSSEC fill the DNS Cache with the same level of trust.

Trust Anchor Manager The database of trust anchors associated to database management processes. This function provides the DNSSEC Validation Engine Trust Anchor information when needed. When TA needs to be updated, the Trust Anchor Manager is also responsible to handle the updating procedure. This includes sending DNS Messages as well as treating appropriately the DNS responses that have been accurately validated by the DNSSEC Validator Engine. This will end up in the DNSSEC Validator Engine pushing new TAs.

DNSSEC Validation Engine follows local policy to approve data. The approved data is returned to the requesting application as well as in the DNS Caches. While the cryptographic computation of the RRSIG signature may be the most visible step, the RRSIG record also contains other information intended to help the validator perform its work, in some cases "sane value" checks are performed. For instance, the original TTL (needed to prepare the RR set for validation) ought to be equal to or higher than the received TTL.

Not shown - Name Server Process Management interfaces to elements, handling of Checking Disabled request, responses, as well as all API requests made of the name server.

5. Time deviation and absence of Real Time Clock

With M2M communication some devices are not expecting to embed Real Time Clock (Raspberry Pi is one example of such devices). When these devices are re-plugged the initial time is set to January 1 1970. Other devices that have clocks that may suffer from time deviation. These devices cannot rely on their time estimation to perform DNSSEC validation.

REQ1 A DNSSEC validator MUST be provided means to update the time without relying on DNSSEC.

Note that updating time in order to be able to perform DNSSEC validation may become a form of a chicken-and-egg problem when the NTP server is designated by its FQDN. The update mechanisms must consider the DNSSEC validator may not able to validate the DNSSEC queries. In other words, the mechanisms may have to update the time over an unsecure DNSSEC resolution.

6. Trust Anchor

6.1. Trust Anchor Bootstrapping

A validator needs to have trust anchors or it will never be able to construct a chain of trust. Trust anchors are defined by DNSSEC to be keys that are inherently trusted, configured by authorized parties, in the validator. The configuration can be via an automated process, such as Automated Updates of DNSSEC Trust Anchors [RFC5011], [I-D.ietf-dnsop-rfc5011-security-considerations], or via manual process.

An implementation of a validator needs to allow an operator to choose any automated process supported by the validator. (No requirements are stated about what processes to support, only one is standardized to date.) An implementation needs to also afford the operator the ability to override or manage via a purely manual process, the storage of managed keys. This includes adding, deleting, changing and inspecting.

Beyond the scope of these requirements are the decision processes of authorized parties in placing trust in keys.

REQ2 A DNSSEC validator MUST check the validity of its Trust Anchors. When a Trust Anchor cannot be verified, the DNSSEC

validator MUST send a warning and SHOULD NOT start validating traffic without manual validation.

REQ3 A DNSSEC validator SHOULD be able to retrieve a Trust Anchor with bootstrapping mechanism. Such mechanism' security MUST NOT be based on DNSSEC, but could instead include downloading a XML file from a trusted URL, or a PKIX certificate.

Although some bootstrapping mechanisms to securely retrieve publish [RFC7958] and retrieve [UNBOUND-ANCHOR] the Root Zone Trust Anchor have been defined, it is believed these mechanisms should be extended to other KSKs or Trust Anchors. In fact it is not always possible to build a trusted delegation between the Root Zone and any sub zone. This may happen for example if one of the upper zones does not handle the secure delegation or improperly implement it. A DS RRset may not be properly filled or its associated signature cannot be validated. As the chain of trust between a zone and the root zone may not be validated, the DNSSEC validation for the zone requires a Trust Anchor. Such DNS(SEC) resolutions may be critical for infrastructure management. A company "Example" may, for example, address all its devices under the domain example.com and may not want disruption to happen if the .com delegation cannot be validated for any reason. Such companies may provision there DNSSEC validator with the Trust Anchor KSK for the zone example.com in addition to the regular DNSSEC delegation. Similarly some some domains may present different views such as a "private" view and a "public view". These zones may have some different content, and may use a different KSK for each view.

6.1.1. The IANA managed root zone KSK

The IANA managed root zone KSK is an operationally significant trust point in the global public Internet. Attention to the trust anchor for this point is paramount. Trust anchor management ought to recognize that the majority of operators deploying DNSSEC validators will need to explicitly or implicitly rely on this trust anchor. Trust anchor management needs to recognize that there may be other trust anchors of interest to operators. Besides deployments in networks other than the global public Internet (hence a different root), operators may want to configure other trust points.

The IANA managed root zone KSK is managed and published as described in "DNSSEC Trust Anchor Publication for the Root Zone" [RFC7598]. That document is written as specific to that trust point. Other trust points may adopt the technique describe (or may use other approaches).

This represents a consideration for implementations. On one hand, operators will place special emphasis on how the root zone DNSSEC KSK

is managed. On the other hand, implementations ought to accommodate trust anchors in a general manner, despite the odds that other trust anchors will not be configured in a specific deployment.

Because of this, it is recommended that implementations make the root zone trust anchor obvious to the operator while still enabling configuration of general trust points.

6.2. Trust Anchor Data Store

When DNSSEC validator are running and a Trust Anchor KSK roll over is ongoing, a network administrator or any trust party may be willing to check whether the new published keys are being stored in a Trust Anchor Data Store with an appropriated status. Such inspection aims at detecting an non successful Trust Anchor roll over before traffic is being rejected. When a new Trust Anchor has not been considered by the DNSSEC validator, a trusted party may be able to provision the DNSSEC validator with the new Trust Anchor, and eventually may remove the revoked Trust Anchor.

While using a Trust Anchor that has been removed results in the DNSSEC validator rejecting multiple legitimate responses, the consequences associated to accepting a rogue Trust Anchor as a legitimate Trust Anchor are even worst. Such attacks would result in an attacker taking control of the entire naming space behind the Trust Anchor. In the case of the Root Zone KSK, for example, almost all name space would be under the control of the attacker. In addition, to the name space, once the rogue Trust Anchor is configured, there is little hope the DNSSEC validator be re-configured with the legitimate Trust Anchor without manual intervention. As a result, it is crucial to cautiously handle operations related to the Trust Anchor provisioning. Means must be provided so network administrator can clearly diagnose the reason a Trust Anchor is not valid to avoid accepting a rogue Trust Anchor inadvertently.

DNSSEC may also be used in some private environment. Corporate networks and home networks, for example, may want to take advantage of DNSSEC for a local scope network. Typically, a corporate network may use a local scope Trust Anchor to validate DNS RRsets provided by authoritative DNSSEC server in the corporate network. This use case is also known as the "split-view" use case. These RRsets within the corporate network may differ from those hosted on the public DNS infrastructure. Note that using different Trust Anchor for a given zone may expose a zone to signature invalidation. This is especially the case for DNSSEC validators that are expected to flip-flop between local and public scope. How validators have to handle the various provisioned Trust Anchors is out of scope of the document.

Home network may use DNSSEC with TLDs or associated domain names that are of local scope and not even registered in the public DNS infrastructure. This requires the ability to manage the Trust Anchor as well.

The necessity to interact with the Trust Anchors lead to the following requirements:

REQ4 A DNSSEC validator MUST store its Trust Anchors in a dedicated Trust Anchor Data Store. Such database MUST store information associated to each Trust Anchor status as well as the time the status has been noticed by the DNSSEC validator. Such database MUST be resilient to DNSSEC validator reboot.

REQ5 Trust Anchor states SHOULD at least consider those described in [RFC5011] (Start, AddPend, Valid, Missing, Revoked, Removed). Additional states SHOULD also be able to indicate additional motivations for revoking the Trust Anchor such as a Trust Anchor known to be corrupted, a Trust anchor miss published, or part of a regular roll over procedure.

REQ6 A DNSSEC validator MUST provide access to the Trust Anchor Data Base to authorized user only. Access control is expected to be based on a least privileged principles.

REQ7 A trusted party MUST be able to add, remove a Trust Anchor in the Trust Anchor Data Store.

6.3. Interactions with the cached RRsets

In addition when a Trust Anchor is revoked, the DNSSEC validator may behave differently if the revocation is motivated by a regular roll over operation or instead by revoking a Trust Anchor that is known as being corrupted. In the case the roll over procedure, is motivated by revoking a Trust Anchor that is known to be corrupted, the DNSSEC validator may be willing to flush all RRsets that depends on the Trust Anchor.

REQ8 A DNSSEC validator MUST be able to flush the cached RRsets that rely on a Trust Anchor.

7. ZSK / KSK

KSK / ZSK are not part of the DNSSEC validator configuration. Their values in the DNS Caches may not reflect those published by the authoritative servers or may be incoherent with the RRset in the DNS Cache they are validating. However, such incoherence primary results from error in the management of the authoritative servers. As a

result, it is not expected that the DNSSEC validator provides complex management facilities to address these issues as this will modify the DNS architecture and add complexity that is not proved to be beneficial.

7.1. KSK/ZSK Data Store

A number of reasons may result in inconsistencies between the RRsets stored in the cache and those published by the authoritative server.

An emergency KSK / ZSK rollover may result in a new KSK / ZSK with associated new RRSIG published in the authoritative zone, while DNSSEC validator may still cache the old value of the ZSK / KSK. For a RRset not cached, the DNSSEC validator performs a DNSSEC query to the authoritative server that returns the RRset signed with the new KSK / ZSK. The DNSSEC validator may not be able to retrieve the new KSK / ZSK while being unable to validate the signature with the old KSK / ZSK. This either result in a bogus resolution or in an invalid signature check. Note that by comparing the Key Tag Fields, the DNSSEC validator is able to notice the new KSK / ZSK used for signing differs from the one used to generate the received generated signature. However, the DNSSEC validator is not expected to retrieve the new ZSK / KSK, as such behavior could be used by an attacker. Instead, ZSK / ZSK key roll over procedure are expected to avoid such inconsistencies.

Similarly, a KSK / ZSK roll over may be performed normally, that is as described in [RFC6781] and [RFC7583]. While the KSK / ZSK roll over is performed, there is no obligation to flush the RRsets in the cache that have been associated with the old key. In fact, these RRset may still be considered as trusted and be removed from the cache as their TTL timeout. With very long TTL, these RRset may remain in the cache while the ZSK / KSK with a shorter TTL is no longer published nor in the cache. In such situations, the purpose of the KSK / ZSK is to validate the data is considered trusted at the time it enters the cache, and such trust may remain after the KSK / ZSK is being rolled over. Note also that even though the data may not be associated to the KSK / ZSK that has been used to validate the data, the link between the KSK / ZSK and the data is still stored in the cache using the RRSIG. Note also that inconsistencies between the ZSK / KSK stored in the cache and those published on the authoritative server, may lead to inconsistencies to downstream DNSSEC validators that rely on multiple cache over time. Typically, a request for the KSK / ZSK may have been provided by a cache that is storing the new published value, while the data and associated signatures may be associated to the old KSK / ZSK.

Incoherence between RRsets and DNSKEYs may be limited by configuring the DNSSEC validator with generic rules that applies to the validation process. Typically, the TTL associate to the DNSKEY is an engagement from the authoritative server that the DNSKEY will remain valid over this period. As this engagement supersedes the validation of any RRSIG and by extension to any RRset in the zone, this TTL value may be used as the maximum value for the TTL associated to FQDNs in the zone. This would at least reduce inconsistencies during regular KSK roll over. In addition, the DNSSEC validator should also be able to provide a maximum values for TTLs.

REQ DNSSEC Validator MUST be able to enforce TTL policies of RRsets based on the TTL of the KSK/ZSK. RRsets TTL SHOULD NOT exceed the KSK / ZSK initial TTL value.

The detection of a misbehaving KSK / ZSK mostly results from publication misconfigurations or an attack at the publication level. As a result, a primary focus is put on DNSSEC Validators monitoring KSK / ZSK with sufficient care to enable the network administrator to take the appropriated actions. Such actions could include out-of-band exchanges as well as specific actions details in section Section 7.2 and section Section 7.3. The monitoring requirements on KSK / ZSK are as follows:

REQ9 A DNSSEC validator MUST log its KSK/ZSK in a dedicated KSK/ ZSK Data Base. Such database MUST store information associated to each KSK/ZSK status as well as the time the status has been noticed by the DNSSEC validator. Such database SHOULD be resilient to DNSSEC validator reboot, that is the information stored in the Data Base MUST NOT be used to populate the cache, while it MAY be used as second factor verification, or audit for example.

REQ10 KSK/ZSK status and information SHOULD be monitored continuously and associated with their respective state as well as verified time. These states and time SHOULD be resilient to reboot.

REQ11 KSK/ZSK states SHOULD at least consider those described in section 3.1 of [RFC7583] (Generated, Published, Ready, Active, Retired, Dead, Removed, Revoked). Additional states SHOULD also be able to indicate additional motivations for revoking the KSK/ ZSK such as a KSK/ZSK known to be corrupted, a KSK/ZSK miss published, or part of a regular roll over procedure.

7.2. KSK ZSK Data Store and Trust Anchor Data Store

A zone may have been badly signed, which means that the KSK or ZSK cannot validate the RRSIG associated to the RRsets. This may not be due to a key roll over, but to an incompatibility between the keys (KSK or ZSK) and the signatures.

When such situation occurs, there is only a choice between not validating the RRsets or invalidating their signature. This is a policy design that needs to be taken by the network administrator. In other ways, flushing the RRset are not expected to address this issue. Such KSK/ZSK are known as Negative Trust Anchors [RFC7646].

With Negative Trust Anchor, the zone for a given time will be known as "known insecure". The DNSSEC Validator is not expected to perform signature validation for this zone. It is expected that this information is associated to a Time To Live (TTL). Note that, this information may be used as an attack vector to impersonate a zone, and must be provided in a trusted way, by a trusted party.

If a zone has been badly signed, the administrator of the authoritative DNS server may resign the zone with the same keys or proceed to an emergency key rollover. If the signature is performed with the same keys, the DNSSEC Validator may notice by itself that RRSIG can be validated. On the other hand if a key rollover is performed, the newly received RRSIG will carry a new key id. Upon receiving a new key id in the RRSIG, the DNSSEC Validator is expected to retrieve the new ZSK/KSK. If the RRSIG can be validated, the DNSSEC validator is expected to remove the "known insecure" flag.

However, if the KSK/ZSK are rolled over and RRSIG cannot be validated, it remains hard for the DNSSEC validator to determine whether the RRSIG cannot be validated or that RRSIG are invalid. As a result:

REQ14 A trusted party MUST be able to indicate a DNSSEC validator that a KSK or a ZSK as Negative Trust Anchor. Such Trust Anchors MUST NOT be used for RRSIG validation and MUST be moved to the Trust Anchor Data Store, so the information become resilient to reboot.

REQ15 A trusted party MUST be able to indicate a DNSSEC validator that a KSK/ZSK is known "back to secure".

7.3. Interactions with cached RRsets

The key roll over procedure intends to ensure that the published RRsets can be validated with the KSK / ZSK stored in the various cache of the DNSSEC validators. As a consequence, the key roll over enables trusted data to be cached. However, the key roll over does not necessarily prevents that cached be always validated with the currently published key. In fact, a cached data may have been validated by the former key and remain in the cache while the former key has been rolled out. Such inconsistencies may be acceptable and correspond to the following trust model: the KSK / ZSK validate the cached data can be trusted at time T. There is no specific information that leads to considers that trust at time T is subject to doubts at current time, so the cached data remain trusted.

While such inconsistencies may have little impact on end host DNSSEC validators, it may be different for large resolving platforms with downstream DNSSEC validators, and a DNSSEC validator may be willing to maintain its cached data consistent with the published KSK / ZSK. A trusted third party may willing to remove all cached RRsets that have been validated by the KSK/ZSK upon some specific states (revoked, or Removed for example), of after some time after the state is noticed. In this later case, only the RRset whose TTL has not expired yet would be flushed.

On the other hand, when a KSK / ZSK is known to be corrupted, this state may affect the trust that has been established at time T. In such case, the DNSSEC validator may be willing to flush all cached data that has been validated by the currently known corrupted KSK / ZSK, including the KSK / ZSK itself.

As a result, the following requirements are expected:

REQ16 A DNSSEC validator MUST be able to flush the cached KSK/ZSK.

REQ17 A DNSSEC validator SHOULD be able to flush the cached RRsets associated to a KSK/ZSK.

8. Cryptography Deprecation

As mentioned in [RFC8247] and [RFC8221] cryptography used one day is expected over the time to be replaced by new and more robust cryptographic mechanisms. In the case of DNSSEC signature protocols are likely to be updated over time. In order to anticipate the sunset of one of the signature scheme, a DNSSEC validator may willing to estimate the impact of deprecating one signature scheme.

Currently [RFC6975] provides the ability for a DNSSEC validator to announce an authoritative server the supported signature schemes. However, a DNSSEC validator is not able to determine other than by trying whether a signature scheme is supported by the authoritative server.

In order for a DNSSEC validator to safely deprecate one signature scheme the following requirement should be fulfilled.

REQ18 A DNSSEC validator SHOULD be able to request the signature scheme supported by an authoritative server.

9. Reporting

A DNSSEC validator receiving a DNS response cannot make the difference between receiving an non-secure response versus an attack. Dropping DNSSEC fields by a misconfigured middle boxes, such as DS, RRRSIG is considered as an attack. A DNSSEC validator is expected to perform secure DNS resolution and as such protect its stub client. An invalid response may be the result of an attack or a misconfiguration, and the DNSSEC validator may play an important role in sharing this information.

REQ19 A DNSSEC validation SHOULD be able to report the unavailability of the DNSSEC service.

REQ20 A DNSSEC validator SHOULD be able to report a invalid DNSSEC validation.

10. IANA Considerations

There are no IANA consideration for this document.

11. Security Considerations

The requirements listed in this document aim at providing the DNSSEC validator appropriated information so DNSSEC validation can be performed. On the other hand, providing inappropriate information can lead to misconfiguring the DNSSEC validator, and thus disrupting the DNSSEC resolution service. As a result, enabling the setting of configuration parameters by a third party may open a wide surface of attacks.

As an appropriate time value is necessary to perform signature check, an attacker may provide rogue time value to prevent the DNSSEC validator to check signatures.

An attacker may also affect the resolution service by regularly asking the DNSSEC validator to flush the KSK/ZSK from its cache. All associated data will also be flushed. This generates additional DNSSEC resolution and additional validations, as RRSet that were cached require a DNSSEC resolution over the Internet. This affects the resolution service by slowing down responses, and increases the load on the DNSSEC validator.

An attacker may ask the DNSSEC validator to consider a rogue KSK/ZSK, thus hijacking the DNS zone. Similarly, an attacker may inform the DNSSEC validator not to trust a given KSK in order to prevent DNSSEC validation to be performed.

An attacker (cf. Section 7) can advertise a "known insecure" KSK or ZSK is "back to secure" to prevent signature check to be performed correctly.

As a result, information considered by the DNSSEC validator should be from a trusted party. This trust party should have been authenticated, and the channel used to exchange the information should also be protected and authenticated.

12. Acknowledgment

The need to address DNSSEC issues on the resolver side started in the Home Networks mailing list and during the IETF87 in Berlin. Among others, people involved in the discussion were Ted Lemon, Ralph Weber, Normen Kowalewski, and Mikael Abrahamsson. People involved in the email discussion initiated by Jim Gettys were, with among others, Paul Wouters, Joe Abley and Michael Richardson.

The current document has been initiated after a discussion with Paul Wouter and Evan Hunt.

13. References

13.1. Normative References

[I-D.ietf-dnsop-rfc5011-security-considerations]
Hardaker, W. and W. Kumari, "Security Considerations for RFC5011 Publishers", draft-ietf-dnsop-rfc5011-security-considerations-13 (work in progress), July 2018.

[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>.

[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>.

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

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

[RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC)", RFC 6975, DOI 10.17487/RFC6975, July 2013, <https://www.rfc-editor.org/info/rfc6975>.

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

[RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J., and R. Weber, "Definition and Use of DNSSEC Negative Trust Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015, <https://www.rfc-editor.org/info/rfc7646>.

[RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <https://www.rfc-editor.org/info/rfc7958>.

[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>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/info/rfc8221>.

[RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <https://www.rfc-editor.org/info/rfc8247>.

13.2. Informative References

[RFC7598] Mrugalski, T., Troan, O., Farrer, I., Perreault, S., Dec, W., Bao, C., Yeh, L., and X. Deng, "DHCPv6 Options for Configuration of Softwire Address and Port-Mapped Clients", RFC 7598, DOI 10.17487/RFC7598, July 2015, <https://www.rfc-editor.org/info/rfc7598>.

[UNBOUND-ANCHOR] "unbound-anchor - Unbound anchor utility", n.d., <https://nlnetlabs.nl/documentation/unbound/unbound-anchor/>.

Authors' Addresses

Daniel Migault
Ericsson
8275 Trans Canada Route
Saint Laurent, QC 4S 0B6
Canada

EMail: daniel.migault@ericsson.com

Edward Lewis
ICANN

EMail: edward.lewis@icann.org

Dan York
ISOC

EMail: york@isoc.org

DNSOP T. Saraj, Ed.
Internet-Draft M. Yousaf
Intended status: Standards Track Riphah Institute of Systems Engineering
Expires: November 12, 2018 A. Qayyum
Capital University of Science and Technology
May 11, 2018

IVIPTR: Resource Record for DNS
draft-tariq-dnsop-iviptr-01

Abstract

This document proposes a new DNS Resource Record IVIPTR which provides the capability to resolve the IPv4 address to IPv6 address and IPv6 address to IPv4 address. This document assumes that the reader is familiar with all the concepts and details discussed in Domain Names Concepts and Facilities [RFC1034] , Domain Names - Implementation and Specification [RFC1035]

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 November 12, 2018.

Table of Contents

1. Introduction

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

The current DNS standard does not support to resolve IPv4 address to IPv6 address and IPv6 address to IPv4 address. For example, if a user program initiate a query for AAAA resource record against an IPv4 address of a domain, the current DNS will return a negative answer normally with RCODE(3)-Non-Existent Domain. Using the current DNS standard, a user program can resolve IPv6 address for a desired IPv4 address by the process as in figure-01:

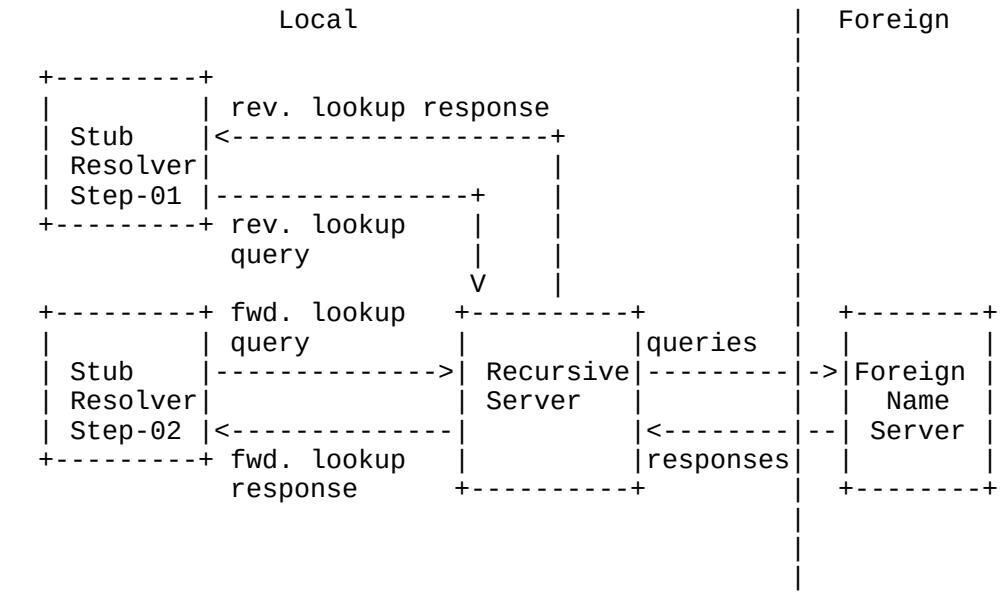


Figure 1

1. The stub-resolver in Step-01, sends a reverse lookup query for an A record to the recursive server to resolve the corresponding fully qualified domain name from the Foreign Name Server

2. The Foreign Name Server returns the PTR resource record against the query to the recursive server, which is responded back to the Stub-resolver as a response.

3. For the received domain name in Step-01, the stub-resolver in Step-02, sends a forward lookup query to recursive server to resolve the corresponding AAAA resource record from the Foreign Name Server.

4. The Foreign Name Server returns the AAAA resource record against the query to the recursive server, which is responded back to the Stub-resolver as a response.

Here, the bottleneck in this process is that now a days, mostly domains has different PTR records for a corresponding A or AAAA resource record. In this case the aforementioned process in figure-01 is not suitable. Also, this process requires to make changes to the Stub-resolver functionality to pursue the aforementioned process. Even, if the Stub-resolver functionality is modified it will work only if a single domain name is used for both A and AAAA record. The proposed solution (IVIPTR) is that, when the

Stub-resolver send a query to the recursive server for resolving AAAA record against an IPv4 address and vice versa, it will respond with the desired resource record (RR) without depending upon a Fully Qualified Domain Name(FQDN) knowledge on Stub-resolver. The term IVI in the proposed IVIPTR resource record is borrowed from one of the IPv4/IPv6 transition mechanisms address translation algorithm [RFC6219].

2. Motivation and Usecases

IPv4 is the principal protocol being used for communication in most of the organizations. Primarily, the need of IVIPTR RR in DNS evolved in a lab environment during the translation of IPv4 security rules to IPv6 security rules in a network security component (Firewall). This section discuss four usecases for the proposed DNS resource record.

2.1. Usecase-01: Firewall Automation

In network security components, mostly traffic monitoring is done through rule based filtering. An organization may enable IPv6 for certain reasons such as:

1. Functionality testing of a newly developed application with IPv6.

2. Performance and compatibility testing of application with IPv6.

3. Or, the organization has decided to keep their network on dual stack from onwards for transition purpose etc.

As a result the security guys has to maintain dual security rules for both Inbound and Outbound network traffic. This can be done by manually configuring the security rules in all network security components for the newly enabled Internet protocol IPv6. Mistakenly, configuring any security rule can result in an undesired consequences.

To automate the security configuration process in a network, there is a need to resolve IPv6 address for a corresponding IPv4 address against every security rule in a network security component (Firewall). The only resource in any network available for this automation process is the DNS. Currently in DNS, there is no such mechanism that can return IPv6 address of a domain if IPv4 address is known or vice versa. The IVIPTR Resource Record conceived as a solution to the problem for resolving IPv6 address if IPv4 address is known or IPv4 address if IPv6 address is known.

There may exist IPv4 or IPv6 address in network security components rules, which does not belong to any fully qualified domain name (FQDN) and thus, are out of the scope of this work. The presence of this IVIPTR Resource Record in the reverse zone file of an authoritative name server can result in automating a number of service for enabling them to reconfigure their security rules for the newly enabled address family protocol i.e. IPv4 or IPv6.

2.2. Usecase-02: Promoting IPv6 Usage

When accessing service such as FTP for a domain say example.com, a user can connect to the server by either:

1. ftp example.com

2. Or, ftp 192.168.0.1

For the second FTP access mechanism, the IVIPTR RR will help to retrieve the IPv6 address against the IPv4 address of the FTP server. Further, the user application will use the newly retrieved IPv6 for connectivity instead of the given one to promote the usage of IPv6 as the priority Internet address for connectivity.

2.3. Usecase-03: Customized Debugging Utilities

Debugging utilities such as traceroute can be customized in such a way that it will give detailed response. For example if a user gives a traceroute command as:

traceroute++ 192.168.0.1 or traceroute++ example.com

Thus, the output will be both PTR record and IVIPTR record.

2.4. Usecase-04: Spam Filtering

When applying spam filtering policy for a mail server such as mail.example.com, the IVIPTR can be helpful in providing additional details such as:

If filtering is performed on IPv4 address, the same can be done for IPv6 address for the corresponding mail server

3. The IVIPTR Resource Record

The IVIPTR RR has mnemonic IVIPTR and type code TBD (decimal). The IVIPTR RR has the following format:

<OWNER> <TTL> <CLASS> IVIPTR <IVI target >

The OWNER is either unqualified or fully qualified domain name depending upon the configuration of reverse zone file optional directive $ORIGIN. The TTL and CLASS fields are the same as for all other PTR records in the reverse zone file. As for the usecases discussed in the previous section the fact of IVIPTR RR usage, it is to be believed that this resource record will not be required to access frequently or in some cases just once, so one can set a smaller TTL value for this resource record to facilitate the recursive server by reducing the cache from unnecessary increase.

IVIPTR is the new RR type that points to a fully qualified domain name (FQDN) i.e. IVI target in a reverse zone file. The <IVI target> from onwards for simplicity written as <target> SHOULD be a fully qualified domain name (FQDN).

The presence of <IVIPTR RR> in a reverse zone can be elaborate by considering the domain example.com. Realistically, most of the times labels in a domain name for an IPv4 and IPv6 glue record are different. There are two possible scenarios for configuration of forward lookup zone file.

3.1. Ideal Scenario

An ideal scenario for a forward lookup zone file would be the one in which, labels in a domain name are same for both IPv4 and IPv6 glue records as:

```
; zone file for example.com

x.example.com.  IN A 192.168.0.1

x.example.com.  IN AAAA 2001:DB8:0::1
```

3.2. Non-Ideal Scenario

A non-ideal scenario for a forward lookup zone file would be the one in which, labels in a domain name are slightly different for both IPv4 and IPv6 glue records as:

```
; zone file for example.com

x.example.com.  IN A 192.168.0.1

x6.example.com.  IN AAAA 2001:DB8:0::1
```

The use of IVIPTR RR is effective only against forward lookup zone file Non-Ideal configuration scenario. Although, it will cause no issue with the Ideal scenario except additional processing overhead.

The representation of IVIPTR RR against both the IPv4 and IPv6 addresses would be as discussed in the next two sub-sections respectively.

3.3. Reverse zone file for IPv4 network prefix

When configuring a reverse zone file for example.com of IPv4 network prefix, the representation of IVIPTR RR type would be as:

```
; reverse zone file example.com for IPv4

1.0.168.192.IN-ADDR.APRPA.  IN PTR x.example.com.

1.0.168.192.IN-ADDR.ARPA.  IN IVIPTR x6.example.com.
```

3.4. Reverse zone file for IPv6 network prefix

When configuring a reverse zone file for example.com of IPv6 network prefix, the representation of IVIPTR RR type would be as:

```
; reverse zone file example.com for IPv6

1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP
6.ARPA.  IN PTR x6.example.com.

1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP
6.ARPA.  IN IVIPTR x.example.com.
```

For the purpose of simplicity in the forward lookup zone file, there is no referral RR Type such as CNAME is listed. In case of presence of any referral record in the forward lookup zone file the <IVI target > in both of the reverse zone files SHOULD be the same as the CNAME < target > in the forward lookup zone file. Thus, IVIPTR MUST follow the rule of robustness principle discussed in section 3.6.2 of [RFC1034] to avoid extra indirections in accessing information.

4. Query Processing

The IVIPTR follow the top level RR format and semantics as defined in the section 3.2.1 of RFC 1035 [RFC1035].

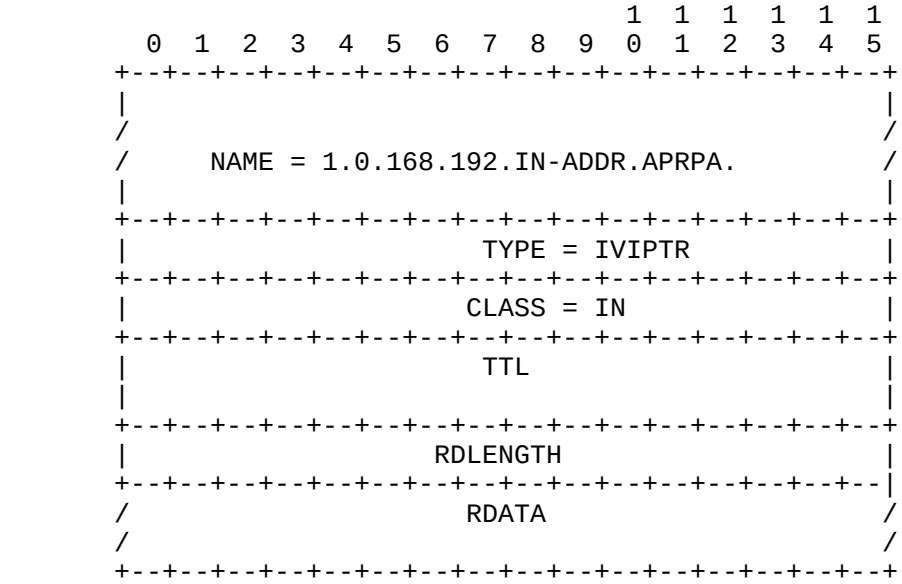


Figure 2

Where:

NAME: the owner name, same as in any reverse lookup query.

TYPE: the two octets field containing the IVIPTR RR TYPE code.

CLASS: two octets containing the RR IN CLASS code value 1.

TTL: the time interval in seconds that the resource record may be cached before the source of the information again to be contacted.

RDLENGTH: specifies the length of RDATA field.

RDATA: A variable length string of octets that represents the <IVI target> resource. The resource depends on the owner in the NAME field of the query.

The query processing is same as any other DNS query except that when the recursive server receives the response for the IVIPTR RR, first it will cache the response like any other resource record and then it will form a new query based on the rules in the sub-sections of this section.

4.1. Client Query: Case-01

If the original query NAME field contains IPv4 representation and TYPE field is IVIPTR then:

1. Upon receiving the response at the recursive server, it SHOULD form a new query.

2. The NAME field of the new query SHOULD be mapped appropriately in the desired format to the IVIPTR target in RDATA resource.

3. The TYPE field for the new query SHOULD be AAAA.

4. This query will be resolved as any other forward lookup query. Upon receiving the response which will contain AAAA RR type target, the recursive server will place this in the answer section of the original query request from client. The IVIPTR RR SHOULD cause no additional section processing.

5. In case of failure or any error the standard error response will be send back to the stub-resolver against the original query request.

4.2. Client Query: Case-02

If the original query NAME field contains IPv6 representation and TYPE field is IVIPTR then:

1. Upon receiving the response at the recursive server, it SHOULD form a new query.

2. The NAME field of the new query SHOULD be mapped appropriately in the desired format to the target in RDATA resource.

3. The TYPE field for the new query SHOULD be A.

4. This query will be resolved as any other forward lookup query. Upon receiving the response which will contain A RR type target, the recursive server will place this in the answer section of the original query request from client. The IVIPTR RR SHOULD cause no additional section processing.

5. In case of failure or any error the standard error response will be send back to the stub-resolver against the original query request.

5. Security Considerations

On a security-aware name server, while resolving the IVIPTR the query processing involves a forward lookup on recursive server in both Section 4.1 and section 4.2 when the new query is formed. The forward lookup in both the cases SHOULD comply completely with the DNSSEC on a security-aware name server and stub-resolver.

6. Acknowledgements

7. IANA Considerations

8. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC6219] Li, X., Bao, C., Chen, M., Zhang, H., and J. Wu, "The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition", RFC 6219, DOI 10.17487/RFC6219, May 2011, <https://www.rfc-editor.org/info/rfc6219>.

Authors' Addresses

Tariq Saraj (editor)
Riphah Institute of Systems Engineering
Aga Khan Road, Sector F-5/1
Islamabad, Federal Capital 44000
Pakistan

Phone: 00923345755556
Email: tariqsaraj@gmail.com

Muhammad Yousaf
Riphah Institute of Systems Engineering
Aga Khan Road, Sector F-5/1
Islamabad, Federal Capital 44000
Pakistan

Email: Muhammad.Yousaf@riu.edu.pk

Amir Qayyum
Capital University of Science and Technology
Kahuta Road, Islamabad Expressway
Islamabad, Federal Capital 44000
Pakistan

Email: aq@cust.edu.pk

DNSOP
Internet-Draft
Intended status: Informational
Expires: January 3, 2018

W. Kumari
Google

July 2, 2017

The .internal TLD.
draft-wkumari-dnsop-internal-00

Abstract

It has become clear that many users would like to use the DNS resolution system for names which do not have meaning in the global context but do have meaning in a context internal to their network. This document reserves the string ".internal" for this purpose.

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

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

[Ed note: This document is intended to drive discussion. It is clear that there has been a desire for an "RFC 1918-style" TLD for a long time; in its absence, people have just started using whatever seemed convenient. This document requests that the allocation of .internal for this use. There is no existing process for this - some of the purpose of this document is to explore the process implications.]

Status of This Memo

This Internet-Draft will expire on January 3, 2018.

Table of Contents

1. Introduction

Over the years, a number of strings have been used as pseudo-TLDs for namespaces that are disjoint (or separate) from the "global DNS" namespace. Common examples of these include .home and .corp. See [I-D.chapin-additional-reserved-tlds], [I-D.ietf-dnsop-sutld-ps] for more background information on this issue. The

[I-D.ietf-dnsop-sutld-ps] document discusses the issues in depth, and should be considered required reading for understanding this document.

The [I-D.ietf-dnsop-alt-tld] document reserves a string to be used as a pseudo-TLD for non-DNS resolution contexts. However, it is clear that there is a significant use case for a similar string to be used for namespaces which are resolved using the DNS protocol, but which do not have a meaning in the global DNS context.

There is no way to prevent users from simply picking a string (such as .home) and starting to use it for internal use. Unfortunately, although these strings are supposed to only be used internally, there is ample evidence that they often leak into the global DNS, sometimes causing technical issues for the user of the no-longer internal name.

This document requests allocation of a string to be used as a pseudo-TLD for namespaces that are not part of the global DNS, but are meant to be resolved with the DNS protocol. A reasonable analogy is that this is to names as [RFC1918] is to IP addresses. Such a reservation should alleviate pollution, such as junk queries at the root, and potential collisions with other users of the namespace.

Note that there was a discussion of the .homenet delegation in the HOMENET WG, and the resulting decision was to *not* request that the name be delegated as a TLD for the IETF's use. This document has significant similarities to the .homenet case, but also significant differences. These include (in no particular order) the fact that the use case is significant broader, and that there is no urgency to the request and does not delay or create uncertainty for any protocol work.

1.1. Requirements notation

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

2. Use cases

The ".internal" TLD can be used for any purpose where a non-global DNS name is needed.

This includes creating a namespace "behind" a Customer Premises Equipment (CPE). For example, the Belkin company used ".belkin" for this. British Telecom used ".home", and shipped devices named "bthomehub.home" [BT_HOME_HUB]). Additionally, internal resolution systems like Microsoft Active Directory have documentation that

suggests (or previously suggested) that administrators use ".corp" for these cases.

The .internal TLD is intended to address some of the issues documented in the Special-Use Domain Names Problem Statement [I-D.ietf-dnsop-sutld-ps] specifically (from the list in Section 3):

5.3: Intended use is covered by gTLD process, but the third party doesn't want to pay a fee.

5.4: Intended use is covered by some IETF process, but the third party doesn't want to follow the process.

5.6: Unaware that a name intended to be used only locally may nevertheless leak

5.7: Unaware that a name used locally with informal allocation may subsequently be allocated formally, creating operational problems.

18 There exists no safe, non-process-violating mechanism for ad-hoc assignment of Special-Use Domain Names.

Other use cases for .internal include its use in testing, prototyping, and benchmarking. Researchers have often needed to set up a fake root and namespace in order to test something, and have needed an arbitrarily chosen a name for a piece of network equipment which it not connected to the Internet.

3. Why not use <existing name>?

The IANA "Special Use Names" registry [IANA.SUN] already contains some names which, it could be argued, already meet this need. Unfortunately, many of these names (such as ".example") are either reserved for a specific use case, or are semantically unsuitable (for example, a CPE manufacturer would likely not find "Open a browser and connect to 'router.invalid'" acceptable).

This section discusses why existing strings in the Special-Use Domain Names registry ([IANA.SUN]) are not appropriate.

3.1. Why not use .alt?

The proposed .alt presudo-TLD is specifically only for use as a pseudo-TLD for non-DNS resolution contexts. At one point .alt was being considered both for DNS and non-DNS resolution contexts, but, after much discussion it was decided that the DNSSEC implications (and desired behavior) meant that .alt should be reserved specifically for non-DNS resolution.

3.2. Why not use something.arpa?

This is indeed an interesting idea. I suspect that it fails the semantically desirable / understandable case, but is definitely worth further discussion. It may also cause issues when server operators override part of the .arpa domain in order to instantiate something.arpa.

3.3. Why not use .local?

.local is already in wide use and has special handling implemented for handing .local names. See [RFC6762] Section 22.1 Subsection 3, 4, 5.

3.4. Why not use .example?

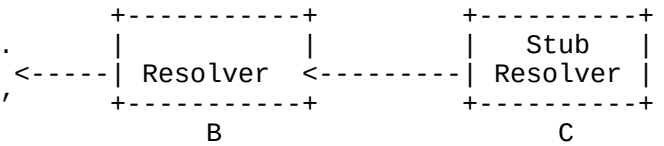
.example, .example.[com|net|org] may indeed be appropriate. Unfortunately, the hard part of all of this is not the selection and adding of a name to the "Special Use Names" registry, but rather deciding if this should be done and, if so, the process to interact with ICANN in order to achieve the delegation.

4. DNSSEC Considerations

The .internal TLD would be an unsigned TLD, as there is no (clean) way to sign it.

This particular topic received much discussion during the "Should .alt be used for DNS, or only non-DNS resolution" discussions, and was the cause of much confusion and misunderstandings. Much of this revolved around why it is important to have an insecure DNSSEC delegation for a name which is added to the Locally-Served DNS Zones ([IANA.LocallyServed]) registry, or any other case where a name is instantiated. It is briefly covered here, but interested readers should review the DNSOP mailing list archive for more details.

Take the following figure:



The user has just purchased a new CPE, which creates an internal namespace called .internal, and responds to lookups for router.interal with its (internal) management IP address (another example would be a corporate user at an organization which has

created a private internal namespace disconnected from the public, global DNS). The CPE hands out addresses using DHCP, and lists itself as the DNS server.

The user follows the instructions included in the box, and enters "http://router.internal" into a web browser. This causes a DNS lookup to be sent from the user's stub to the recursive resolver. The recursive resolver correctly identifies that this is query is for itself, and so responds with an answer saying "router.internal is 192.168.0.1".

4.1. Scenario 1 - No DNSSEC, .internal not delegated

In this scenario, the .internal name has not been added to the root zone, and the user's stub resolver *does not* perform DNSSEC validation. The stub receives the response, performs no validation, and so trust the answer and connects. The user is happy. This is the current behavior for non-DNSSEC validating stubs.

4.2. Scenario 2 - DNSSEC, .internal not delegated

In this scenario, the .internal name has not been added to the root zone, and the user's stub resolver *does* perform DNSSEC validation. (Note that it is believed that validating stubs are currently rare.) The stub receives the response, and begins DNSSEC validation. As the .internal TLD has not been added to the root zone, DNSSEC Authenticated Denial of Existence proves that the .internal TLD does not exist (currently there is an NSEC record proving that nothing exists between .intel and .international). This resulting outcome is a DNSSEC "bogus" answer, the user is unable to connect, and becomes frustrated. This is the current behavior for DNSSEC validating stubs.

4.3. Scenario 3 - DNSSEC, .internal delegated

In this scenario, the .internal name has been added to the DNS root zone, with an insecure delegation to AS112 (by "insecure delegation" we mean that that there is no DS record for .internal in the root zone; the .internal domain is unsigned). The stub receives the response, and performs DNSSEC validation. As .internal has been delegated, there is an (insecure) entry in the root zone, proving that the .internal TLD exists. As it is an insecure delegation, the validating stub is perfectly happy to accept the "router.internal is 192.168.0.1" response, the user connects to their router, and everyone is happy.

5. IANA Considerations

This document requests that the .internal TLD be assigned to the IANA (similar to the way that .example is) and a DNSSEC insecure delegation (that is, a delegation with no DS records) be inserted into the root-zone, delegated to blackhole-[12].iana.org.

[Editor's note: This is not something which the IANA currently has the authority to do. This fact was extensively discussed during the .homenet discussions. This text in the IANA considerations should be considered a placeholder for "what someone needs to do if this gets IETF consensus". If there is consensus that the reservation of .internal makes sense, there will need to be some process design before implementation. Such a process might be similar to "the IESG asks the IAB to request ICANN to consider the reservation / delegation of this string". ICANN does not currently have a process for handling requests like these, and so will also likely need to design a process for this. It is possible that either process might not be designed and this will fail. It is also possible that the IETF makes a request and ICANN does not make the delegation.]

5.1. Domain Name Reservation Considerations

[Ed: This section is to satisfy the requirement in Section 5 of RFC6761. This document is intended to spark a discussion, there is currently no process for the IETF / IAB to request that ICANN delegate a TLD for special use, and simply adding .internal to the "Special-Use Domain Name" registry ([IANA.SUN]) does not accomplish this. I have decided to fill in the RFC6761 "questions" simply to clarify the expected behavior. This entire section needs to be removed before publication.]

The string ".internal." is special in the following ways:

1. Users may know that strings that end in .internal behave differently to normal DNS names. They may expect that names of the form <something>.internal refer to resources internal to their network / enterprise / similar.

2. Writers of application software not required to perform any special handing for .internal names. They are resolved normally, using the DNS.

3. Writers of name resolution APIs and libraries are not expected to perform special handling.

4. Caching DNS servers MAY recognize these names as special. By default they should answer them locally (using "Locally Served

Zones"), but may be configured to forward them to authoritative servers.

5. Authoritative DNS servers SHOULD NOT recognize these names as special and should not perform any special handling with them, unless they wish to instantiate an internal namespace, in which case they can choose to simply create any names within .internal that they want. Names are "local" to the network, and only have meaning within that network.

6. DNS server operators SHOULD be aware that queries for names ending in .internal are not part of the global, IANA DNS, and were leaked into the global DNS. This information may be useful for support or debugging purposes.

7. DNS Registries/Registrars MUST NOT grant requests to register ".internal" names in the normal way to any person or entity. These ".internal" names are defined by protocol specification to be nonexistent, and they fall outside the set of names available for allocation by registries/registrars.

6. Security Considerations

This section will certainly be filled in later as the discussion progresses.

7. Acknowledgements

The authors wish to thank Steve Crocker, Wes Hardaker, David Lawrence, Suzanne Woolf, and many others on the DNSOP mailing list.

The author would like to especially think Paul Hoffman for creating a Pull Request.

8. References

8.1. Normative References

[I-D.ietf-dnsop-sutld-ps]
Lemon, T., Droms, R., and W. Kumari, "Special-Use Domain Names Problem Statement", draft-ietf-dnsop-sutld-ps-06 (work in progress), June 2017.

[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>.

8.2. Informative References

[BT_HOME_HUB]
IANA, "I have problems connecting 5GHz and dual band devices wirelessly to the BT Hub", <http://bt.custhelp.com/app/answers/detail/a_id/44798/kw/bthomehub.home/c/346,7474,7477>.

[I-D.chapin-additional-reserved-tlds]
Chapin, L. and M. McFadden, "Additional Reserved Top Level Domains", draft-chapin-additional-reserved-tlds-02 (work in progress), March 2015.

[I-D.ietf-dnsop-alt-tld]
Kumari, W. and A. Sullivan, "The ALT Special Use Top Level Domain", draft-ietf-dnsop-alt-tld-08 (work in progress), March 2017.

[IANA.LocallyServed]
IANA, "Locally Served DNS Zones", <https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>.

[IANA.SUN]
IANA, "Special-Use Domain Names", <https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

-00

- This document was originally started in late 2014 / early 2015, but languished until being revived in June 2017.

Author's Address

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

Email: warren@kumari.net