dprive
Internet-Draft
Updates: 1995, 7766 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

W. Toorop
NLnet Labs
S. Dickinson
Sinodun IT
S. Sahib
P. Aras
A. Mankin
Salesforce

July 13, 2020

DNS Zone Transfer-over-TLS
draft-ietf-dprive-xfr-over-tls-02

Abstract

DNS zone transfers are transmitted in clear text, which gives attackers the opportunity to collect the content of a zone by eavesdropping on network connections. The DNS Transaction Signature (TSIG) mechanism is specified to restrict direct zone transfer to authorized clients only, but it does not add confidentiality. This document specifies use of TLS, rather then clear text, to prevent zone contents collection via passive monitoring of zone transfers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DNS has a number of privacy vulnerabilities, as discussed in detail in [RFC7626]. Stub client to recursive resolver query privacy has received the most attention to date, with standards track documents for both DNS-over-TLS (DoT) [RFC7858] and DNS-over-HTTPS (DoH) [RFC8484], and a proposal for DNS-over-QUIC [I-D.ietf-dprive-dnsoquic]. There is ongoing work on DNS privacy requirements for exchanges between recursive resolvers and authoritative servers [I-D.ietf-dprive-phase2-requirements] and some suggestions for how signaling of DoT support by authoritatives might work, e.g., [I-D.vandijk-dprive-ds-dot-signal-and-pin]. However there is currently no RFC that specifically defines authoritative support for DNS-over-TLS.

[RFC7626] established that stub client DNS query transactions are not public and needed protection, but on zone transfer [RFC1995] [RFC5936] it says only:

> "Privacy risks for the holder of a zone (the risk that someone gets the data) are discussed in [RFC5936] and [RFC5155]."

In what way is exposing the full contents of a zone a privacy risk? The contents of the zone could include information such as names of persons used in names of hosts. Best practice is not to use personal information for domain names, but many such domain names exist. The contents of the zone could also include references to locations that allow inference about location information of the individuals associated with the zone's organization. It could also include references to other organizations. Examples of this could be:

- o Person-laptop.example.org

- MX-for-Location.example.org

- Service-tenant-from-another-org.example.org

There may also be regulatory, policy or other reasons why the zone contents in full must be treated as private.

Neither of the RFCs mentioned in [RFC7626] contemplates the risk that someone gets the data through eavesdropping on network connections, only via enumeration or unauthorized transfer as described in the following paragraphs.

[RFC5155] specifies NSEC3 to prevent zone enumeration, which is when queries for the authenticated denial of existences records of DNSSEC allow a client to walk through the entire zone. Note that the need for this protection also motivates NSEC5 [I-D.vcelak-nsec5]; zone walking is now possible with NSEC3 due to crypto-breaking advances, and NSEC5 is a response to this problem.

[RFC5155] does not address data obtained outside zone enumeration (nor does [I-D.vcelak-nsec5]). Preventing eavesdropping of zone transfers (this draft) is orthogonal to preventing zone enumeration, though they aim to protect the same information.

[RFC5936] specifies using TSIG [RFC2845] for authorization of the clients of a zone transfer and for data integrity, but does not express any need for confidentiality, and TSIG does not offer encryption. Some operators use SSH tunneling or IPSec to encrypt the transfer data.

Because both AXFR and IXFR zone transfers are typically carried out over TCP from authoritative DNS protocol implementations, encrypting zone transfers using TLS, based closely on DoT [RFC7858], seems like a simple step forward. This document specifies how to use TLS as a transport to prevent zone collection from zone transfers.

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] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Privacy terminology is as described in Section 3 of [RFC6973].

Note that in this document we choose to use the terms 'primary' and 'secondary' for two servers engaged in zone transfers.

DNS terminology is as described in [RFC8499].

DoT: DNS-over-TLS as specified in [RFC7858]

XoT: Generic XFR-over-TLS mechanisms as specified in this document

AXoT: AXFR-over-TLS

IXoT: IXFR over-TLS

3. Use Cases for XFR-over-TLS

- Confidentiality. Clearly using an encrypted transport for zone transfers will defeat zone content leakage that can occur via passive surveillance.

- Authentication. Use of single or mutual TLS authentication (in combination with ACLs) can complement and potentially be an alternative to TSIG.

- Performance. Existing AXFR and IXFR mechanisms have the burden of backwards compatibility with older implementations based on the original specifications in [RFC1034] and [RFC1035]. For example, some older AXFR servers don't support using a TCP connection for multiple AXFR sessions or XFRs of different zones because they have not been updated to follow the guidance in [RFC5936]. Any implementation of XFR-over-TLS (XoT) would obviously be required to implement optimized and interoperable transfers as described in [RFC5936], e.g., transfer of multiple zones over one connection.

- Performance. Current usage of TCP for IXFR is sub-optimal in some cases i.e. connections are frequently closed after a single IXFR.

4. Connection and Data Flows in Existing XFR Mechanisms

The original specification for zone transfers in [RFC1034] and [RFC1035] was based on a polling mechanism: a secondary performed a periodic SOA query (based on the refresh timer) to determine if an AXFR was required.

[RFC1995] and [RFC1996] introduced the concepts of IXFR and NOTIFY respectively, to provide for prompt propagation of zone updates. This has largely replaced AXFR where possible, particularly for dynamically updated zones.

[RFC5936] subsequently redefined the specification of AXFR to improve performance and interoperability.

In this document we use the phrase "XFR mechanism" to describe the entire set of message exchanges between a secondary and a primary that concludes in a successful AXFR or IXFR request/response. This set may or may not include

- NOTIFY messages
- SOA queries
- Fallback from IXFR to AXFR
- Fallback from IXFR-over-UDP to IXFR-over-TCP

The term is used to encompasses the range of permutations that are possible and is useful to distinguish the 'XFR mechanism' from a single XFR request/response exchange.

4.1. AXFR Mechanism

The figure below provides an outline of an AXFR mechanism including NOTIFYs.

Figure 1. AXFR Mechanism [1]

1. An AXFR is often (but not always) preceded by a NOTIFY (over UDP) from the primary to the secondary. A secondary may also initiate an AXFR based on a refresh timer or scheduled/triggered zone maintenance.

2. The secondary will normally (but not always) make a SOA query to the primary to obtain the serial number of the zone held by the primary.

3. If the primary serial is higher than the secondaries serial (using Serial Number Arithmetic [RFC1982]), the secondary makes an AXFR request (over TCP) to the primary after which the AXFR data flows in one or more AXFR responses on the TCP connection.

[RFC5936] specifies that AXFR must use TCP as the transport protocol but details that there is no restriction in the protocol that a single TCP connection must be used only for a single AXFR exchange, or even solely for XFRs. For example, it outlines that the SOA query can also happen on this connection. However, this can cause interoperability problems with older implementations that support only the trivial case of one AXFR per connection.

Further details of the limitations in existing AXFR implementations are outlined in [RFC5936].

4.2. IXFR Mechanism

The figure below provides an outline of the IXFR mechanism including NOTIFYs.

Figure 1. IXFR Mechanism [2]

1. An IXFR is normally (but not always) preceded by a NOTIFY (over UDP) from the primary to the secondary. A secondary may also initiate an IXFR based on a refresh timer or scheduled/triggered zone maintenance.

2. The secondary will normally (but not always) make a SOA query to the primary to obtain the serial number of the zone held by the primary.

3. If the primary serial is higher than the secondaries serial (using Serial Number Arithmetic [RFC1982]), the secondary makes an IXFR request to the primary after the primary sends an IXFR response.

[RFC1995] specifies that Incremental Transfer may use UDP if the entire IXFR response can be contained in a single DNS packet, otherwise, TCP is used. In fact is says in non-normative language:

> "Thus, a client should first make an IXFR query using UDP."

So there may be a forth step above where the client falls back to IXFR-over-TCP. There may also be a forth step where the secondary must fall back to AXFR because, e.g., the primary does not support IXFR.

However it is noted that at least two widely used open source authoritative nameserver implementations (BIND [3] and NSD [4]) do IXFR using TCP by default in their latest releases. For BIND TCP connections are sometimes used for SOA queries but in general they are not used persistently and close after an IXFR is completed.

It is noted that the specification for IXFR was published well before TCP was considered a first class transport for DNS. This document therefore updates [RFC1995] to state that DNS implementations that support IXFR-over-TCP MUST use [RFC7766] to optimize the use of TCP connections and SHOULD use [RFC7858] to manage persistent connections.

4.3. Data Leakage of NOTIFY and SOA Message Exchanges

This section attempts to presents a rationale for also encrypting the other messages in the XFR mechanism.

Since the SOA of the published zone can be trivially discovered by simply querying the publicly available authoritative servers leakage of this RR is not discussed in the following sections.

4.3.1. NOTIFY

Unencrypted NOTIFY messages identify configured secondaries on the primary.

[RFC1996] also states:

> "If ANCOUNT>0, then the answer section represents an unsecure hint at the new RRset for this (QNAME,QCLASS,QTYPE).

But since the only supported QTYPE for NOTIFY is SOA, this does not pose a potential leak.

4.3.2. SOA

For hidden primaries or secondaries the SOA response leaks the degree of lag of any downstream secondary.

5. Connections and Data Flows in XoT

5.1. TLS versions

For improved security all implementations of this specification MUST use only TLS 1.3 [RFC8446] or later.

5.2. Connection usage

It is useful to note that in these mechanisms it is the secondary that initiates the TLS connection to the primary for a XFR request, so that in terms of connectivity the secondary is the TLS client and the primary the TLS server.

The details in [RFC7766], [RFC7858] and [RFC8310] about, e.g., persistent connection and message handling are fully applicable to XoT as well. However any behavior specified here takes precedence for XoT.

5.2.1. High level XoT descriptions

The figure below provides an outline of the AXoT mechanism including NOTIFYs.

Figure 3: AXoT mechanism [5]

The figure below provides an outline of the IXoT mechanism including NOTIFYs.

Figure 4: IXoT mechanism [6]

5.2.2. Previous specifications

We note that whilst [RFC5936] already recommends re-using open TCP connections, it does state:

> "Non-AXFR session traffic can also use an open TCP connection."

when discussing AXFR-over-TCP. It defines an AXFR session as an AXFR query message and the sequence of AXFR response messages returned for it. Note that this excludes any SOA queries issued as part of the overall AXFR mechanism. This requirement needs to be re-evaluated when considering applying the same model to XoT since

- There is no guarantee that a XoT server (which is very likely, but not necessarily, a purely authoritative server) will also support DoT for regular queries. Requiring a purely authoritative server to also respond to any query over a TLS connection would be equivalent to defining a form of authoritative DoT. We consider this to be out of scope for this document, which is focussed purely on zone transfers.

- It would, however, be optimal for XoT to include the capability to send SOA queries over an already open TLS connection.

Moreover, it is worth noting that [RFC7766] made general implementation recommendations with regard to TCP/TLS connection handling:

> "To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimize the number of concurrent TCP connections made to any individual server. It is RECOMMENDED that for any given client/server interaction there SHOULD be no more than one connection for regular queries, one for zone transfers, and one for each protocol that is being used on top of TCP (for example, if the resolver was using TLS). However, it is noted that certain primary/ secondary configurations with many busy zones might need to use more than one TCP connection for zone transfers for operational reasons (for example, to support concurrent transfers of multiple zones)."

Whilst this recommends a particular behavior for the clients using TCP, it does not relax the requirement for servers to handle 'mixed' traffic (regular queries and zone transfers) on any open TCP/TLS connection. It also overlooks the potential that other transports might want to take the same approach with regard to using separate connections for different purposes.

5.3. Update to RFC7766

This specification for XoT updates the guidance in [RFC7766] to provide the same separation of connection purpose (regular queries and zone transfers) for all transports being used on top of TCP. Therefore, it is RECOMMENDED that for each protocol used on top of TCP in any given client/server interaction there SHOULD be no more than one connection for regular queries and one for zone transfers. We provide specific details in the following sections of reasons where more than one connection might be required for zone transfers.

5.4. Connection Establishment

This specification additionally limits the scope of XoT as defined here to be the use of dedicated TLS connections (XoT connections) to exchange only traffic specific to enabling zone transfers. The set of transactions supported on such connections is limited to:

- AXFR
- IXFR
- SOA

and is collectively referred to hereafter as 'XoT traffic'.

Such connections MUST use an ALPN token of 'xot' during the TLS handshake (see Section 11).

In the absence of DNS specific capability signaling mechanisms this greatly simplifies the implementation of XoT such that a XoT exchange can occur between any primary and secondary regardless of the role of each (e.g. purely authoritative, recursive resolver also authoritatively hosting zones, stub) or of other DNS transport capability each may have. It also clearly makes XoT support orthogonal to any set of zone transfer authentication mechanisms chosen by the two parties.

XoT clients MUST only send XoT traffic on XoT connections. If a XoT server receives traffic other than XoT traffic on a XoT connection it MUST respond with the extended DNS error code 21 - Not Supported [I-D.ietf-dnsop-extended-error]. It SHOULD treat this as protocol error and close the connection.

With the update to [RFC7766] guidance above, clients are free to open separate connections to the server to make any other queries they may need over either TLS, TCP or UDP. A specification for connections that support both XoT traffic and non-XoT traffic may be the subject of a future work.

5.4.1. Draft Version Identification

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Only implementations of the final, published RFC can identify themselves as "xot". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier. For example, draft-ietf-dprive-xfr-over-tls-02 is identified using the string "xot-02".

5.5. Port selection

The connection for XoT SHOULD be established using port 853, as specified in [RFC7858], unless there is mutual agreement between the secondary and primary to use a port other than port 853 for XoT. There MAY be agreement to use different ports for AXoT and IXoT.

5.6. AXoT mechanism

5.6.1. Coverage and relationship to RFC5936

[RFC5936] re-specified AXFR providing additional guidance beyond that provided in [RFC1034] and [RFC1035]. For example, sections 4.1, 4.1.1 and 4.1.2 of [RFC5936] provide improved guidance for AXFR clients and servers with regard to re-use of connections for multiple AXFRs and AXFRs of different zones. However [RFC5936] was constrained by having to be backwards compatible with some very early basic implementations of AXFR.

Here we specify some optimized behaviors for AXoT, based closely on those in [RFC5936], but without the constraint of backwards compatibility since it is expected that all implementations of AXoT fully implement the behavior described here.

Where any behavior is not explicitly described here, the behavior specified in [RFC5936] MUST be followed. Any behavior specified here takes precedence for AXoT implementations over that in [RFC5936].

5.6.2. AXoT connection and message handling

The first paragraph of Section 4.1.1 of [RFC5936] says that clients SHOULD close the connection when there is no 'apparent need' to use the connection for some time period.

For AXoT this requirement is updated: AXoT clients and servers SHOULD use EDNS0 Keepalive [RFC7828] to establish the connection timeouts to be used. The client SHOULD send the EDNS0 Keepalive option on every AXoT request sent so that the server has every opportunity to update the Keepalive timeout. The AXoT server may use the frequency of recent AXFRs to calculate an average update rate as input to the decision of what EDNS0 Keepalive timeout to use. If the server does not support EDNS0 Keepalive the client MAY keep the connection open for a few seconds ([RFC7766] recommends that servers use timeouts of at least a few seconds).

Whilst the specification for EDNS0 [RFC6891] does not specifically mention AXFRs, it does say

> "If an OPT record is present in a received request, compliant responders MUST include an OPT record in their respective responses."

We clarify here that if an OPT record is present in a received AXoT request, compliant responders MUST include an OPT record in each of the subsequent AXoT responses. Note that this requirement, combined with the use of EDNS0 Keepalive, enables AXoT servers to signal the desire to close a connection due to low resources by sending an EDNS0

Keepalive option with a timeout of 0 on any AXoT response (in the absence of another way to signal the abort of a AXoT transfer).

An AXoT server MUST be able to handle multiple AXFR requests on a single XoT connection (for the same and different zones).

[RFC5936] says:

> "An AXFR client MAY use an already opened TCP connection to start an AXFR session. Using an existing open connection is RECOMMENDED over opening a new connection. (Non-AXFR session traffic can also use an open connection.)"

For AXoT this requirement is updated: AXoT clients SHOULD re-use an existing open XoT connection when starting any new AXoT session to the same primary, and for issuing SOA queries, instead of opening a new connection. The number of XoT connections between a secondary and primary SHOULD be minimized.

Valid reasons for not re-using existing connections might include:

- o reaching a configured limit for the number of outstanding queries allowed on a single XoT connection

- o the message ID pool has already been exhausted on an open connection

- o a large number of timeouts or slow responses have occurred on an open connection

- o an EDNS0 Keepalive option with a timeout of 0 has been received from the server and the client is in the process of closing the connection

If no XoT connections are currently open, AXoT clients MAY send SOA queries over UDP, TCP or TLS.

[RFC5936] says:

> "Some old AXFR clients expect each response message to contain only a single RR. To interoperate with such clients, the server MAY restrict response messages to a single RR."

This is opposed to the normal behavior of containing a sufficient number of RRs to reasonably amortize the per-message overhead. We clarify here that AXoT clients MUST be able to handle responses that include multiple RRs, up to the largest number that will fit within a DNS message (taking the required content of the other sections into account, as described here and in [RFC5936]). This removes any burden on AXoT servers of having to accommodate a configuration option or support for restricting responses to containing only a single RR.

An AXoT client SHOULD pipeline AXFR requests for different zones on a single XoT connection. An AXoT server SHOULD respond to those requests as soon as the response is available i.e. potentially out of order.

5.6.3. Padding AXoT responses

The goal of padding AXoT responses would be two fold:

- to obfuscate the actual size of the transferred zone to minimize information leakage about the entire contents of the zone.

- to obfuscate the incremental changes to the zone between SOA updates to minimize information leakage about zone update activity and growth.

Note that the re-use of XoT connections for transfers of multiple different zones complicates any attempt to analyze the traffic size and timing to extract information.

We note here that any requirement to obfuscate the total zone size is likely to require a server to create 'empty' AXoT responses. That is, AXoT responses that contain no RR's apart from an OPT RR containing the EDNS(0) option for padding. However, as with existing AXFR, the last AXoT response message sent MUST contain the same SOA that was in the first message of the AXoT response series in order to signal the conclusion of the zone transfer.

[RFC5936] says:

> "Each AXFR response message SHOULD contain a sufficient number of RRs to reasonably amortize the per-message overhead, up to the largest number that will fit within a DNS message (taking the required content of the other sections into account, as described below)."

'Empty' AXoT responses generated in order to meet a padding requirement will be exceptions to the above statement. In order to guarantee support for future padding policies, we state here that secondary implementations MUST be resilient to receiving padded AXoT responses, including 'empty' AXoT responses that contain only an OPT RR containing the EDNS(0) option for padding.

Recommendation of specific policies for padding AXoT responses are out of scope for this specification. Detailed considerations of such policies and the trade-offs involved are expected to be the subject of future work.

5.7. IXoT mechanism

5.7.1. Coverage and relationship to RFC1995

[RFC1995] says nothing with respect to optimizing IXFRs over TCP or re-using already open TCP connections to perform IXFRs or other queries. Therefore, there arguably is an implicit assumption (probably unintentional) that a TCP connection is used for one and only one IXFR request. Indeed, several open source implementations currently take this approach.

We provide new guidance here specific to IXoT that aligns with the guidance in [RFC5936] for AXFR, that in section Section 5.6 for AXoT, and with that for performant TCP/TLS usage in [RFC7766] and [RFC7858].

Where any behavior is not explicitly described here, the behavior specified in [RFC1995] MUST be followed. Any behavior specified here takes precedence for IXoT implementations over that in [RFC1995].

5.7.2. IXoT connection and message handling

In a manner entirely analogous to that described in paragraph 2 of Section 5.6.2 IXoT clients and servers SHOULD use EDNS0 Keepalive [RFC7828] to establish the connection timeouts to be used.

An IXoT server MUST be able to handle multiple IXoT requests on a single XoT connection (for the same and different zones).

IXoT clients SHOULD re-use an existing open XoT connection when making any new IXoT request to the same primary, and for issuing SOA queries, instead of opening a new connection. The number of XoT connections between a secondary and primary SHOULD be minimized.

Valid reasons for not re-using existing connections are the same as those described in Section 5.6.2

If no XoT connections are currently open, IXoT clients MAY send SOA queries over UDP, TCP or TLS.

An IXoT client SHOULD pipeline IXFR requests for different zones on a single XoT connection. An IXoT server SHOULD respond to those requests as soon as the response is available i.e. potentially out of order.

5.7.3. Condensation of responses

[RFC1995] says condensation of responses is optional and MAY be done. Whilst it does add complexity to generating responses it can significantly reduce the size of responses. However any such reduction might be offset by increased message size due to padding. This specification does not update the optionality of condensation.

5.7.4. Fallback to AXFR

Fallback to AXFR can happen, for example, if the server is not able to provide an IXFR for the requested SOA. Implementations differ in how long they store zone deltas and how many may be stored at any one time.

After a failed IXFR a IXoT client SHOULD request the AXFR on the already open XoT connection.

5.7.5. Padding of IXoT responses

The goal of padding IXoT responses would be to obfuscate the incremental changes to the zone between SOA updates to minimize information leakage about zone update activity and growth. Both the size and timing of the IXoT responses could reveal information.

IXFR responses can vary in size greatly from the order of 100 bytes for one or two record updates, to tens of thousands of bytes for large dynamic DNSSEC signed zones. The frequency of IXFR responses can also depend greatly on if and how the zone is DNSSEC signed.

In order to guarantee support for future padding policies, we state here that secondary implementations MUST be resilient to receiving padded IXoT responses.

Recommendation of specific policies for padding IXoT responses are out of scope for this specification. Detailed considerations of such policies and the trade-offs involved are expected to be the subject of future work.

6. Multi-primary Configurations

Also known as multi-master configurations this model can provide flexibility and redundancy particularly for IXFR. A secondary will receive one or more NOTIFY messages and can send an SOA to all of the configured primaries. It can then choose to send an XFR request to the primary with the highest SOA (or other criteria, e.g., RTT).

When using persistent connections the secondary may have a XoT connection already open to one or more primaries. Should a secondary preferentially request an XFR from a primary to which it already has an open XoT connection or the one with the highest SOA (assuming it doesn't have a connection open to it already)?

Two extremes can be envisaged here. The first one can be considered a 'preferred primary connection' model. In this case the secondary continues to use one persistent connection to a single primary until it has reason not to. Reasons not to might include the primary repeatedly closing the connection, long RTTs on transfers or the SOA of the primary being an unacceptable lag behind the SOA of an alternative primary.

The other extreme can be considered a 'parallel primary connection' model. Here a secondary could keep multiple persistent connections open to all available primaries and only request XFRs from the primary with the highest serial number. Since normally the number of secondaries and primaries in direct contact in a transfer group is reasonably low this might be feasible if latency is the most significant concern.

Recommendation of a particular scheme is out of scope of this document but implementations are encouraged to provide configuration options that allow operators to make choices about this behavior.

7. Zone Transfer with DoT - Authentication

7.1. TSIG

TSIG [RFC2845] provides a mechanism for two or more parties to use shared secret keys which can then be used to create a message digest to protect individual DNS messages. This allows each party to authenticate that a request or response (and the data in it) came from the other party, even if it was transmitted over an unsecured channel or via a proxy. It provides party-to-party data authentication, but not hop-to-hop channel authentication or confidentiality.

7.2. SIG(0)

SIG(0) [RFC2535] similarly also provides a mechanism to digitally sign a DNS message but uses public key authentication, where the public keys are stored in DNS as KEY RRs and a private key is stored at the signer. It also provides party-to-party data authentication, but not hop-to-hop channel authentication or confidentiality.

7.3. TLS

7.3.1. Opportunistic

Opportunistic TLS [RFC8310] provides a defense against passive surveillance, providing on-the-wire confidentiality.

7.3.2. Strict

Strict TLS [RFC8310] requires that a client is configured with an authentication domain name (and/or SPKI pinset) that should be used to authenticate the TLS handshake with the server. This additionally provides a defense for the client against active surveillance, providing client-to-server authentication and end-to-end channel confidentiality.

7.3.3. Mutual

This is an extension to Strict TLS [RFC8310] which requires that a client is configured with an authentication domain name (and/or SPKI pinset) and a client certificate. The client offers the certificate for authentication by the server and the client can authentic the server the same way as in Strict TLS. This provides a defense for both parties against active surveillance, providing bi-directional authentication and end-to-end channel confidentiality.

7.4. IP Based ACL on the Primary

Most DNS server implementations offer an option to configure an IP based Access Control List (ACL), which is often used in combination with TSIG based ACLs to restrict access to zone transfers on primary servers.

This is also possible with XoT but it must be noted that as with TCP the implementation of such an ACL cannot be enforced on the primary until a XFR request is received on an established connection.

If control were to be any more fine-grained than this then a separate, dedicated port would need to be agreed between primary and secondary for XoT such that implementations would be able to refuse connections on that port to all clients except those configured as secondaries.

7.5. ZONEMD

Message Digest for DNS Zones (ZONEMD) [I-D.ietf-dnsop-dns-zone-digest] digest is a mechanism that can be used to verify the content of a standalone zone. It is designed to be independent of the transmission channel or mechanism, allowing a general consumer of a zone to do origin authentication of the entire zone contents. Note that the current version of [I-D.ietf-dnsop-dns-zone-digest] states:

"As specified at this time, ZONEMD is not designed for use in large, dynamic zones due to the time and resources required for digest calculation. The ZONEMD record described in this document includes fields reserved for future work to support large, dynamic zones."

It is complementary the above mechanisms and can be used in conjunction with XoT but is not considered further.

7.6. Comparison of Authentication Methods

The Table below compares the properties of a selection of the above methods in terms of what protection they provide to the secondary and primary servers during XoT in terms of:

- 'Data Auth': Authentication that the DNS message data is signed by the party with whom credentials were shared (the signing party may or may not be party operating the far end of a TCP/TLS connection in a 'proxy' scenario). For the primary the TSIG on the XFR request confirms that the requesting party is authorized to request zone data, for the secondary it authenticates the zone data that is received.

- 'Channel Conf': Confidentiality of the communication channel between the client and server (i.e. the two end points of a TCP/TLS connection).

- Channel Auth: Authentication of the identity of party to whom a TCP/TLS connection is made (this might not be a direct connection between the primary and secondary in a proxy scenario).

It is noted that zone transfer scenarios can vary from a simple single primary/secondary relationship where both servers are under the control of a single operator to a complex hierarchical structure which includes proxies and multiple operators. Each deployment scenario will require specific analysis to determine which authentication methods are best suited to the deployment model in question.

Table 1: Properties of Authentication methods for XoT [7]

Based on this analysis it can be seen that:

- A combination of Opportunistic TLS and TSIG provides both data authentication and channel confidentiality for both parties. However this does not stop a MitM attack on the channel which could be used to gather zone data.

- Using just mutual TLS can be considered a standalone solution if the secondary has reason to place equivalent trust in channel authentication as data authentication, e.g., the same operator runs both the primary and secondary.

- Using TSIG, Strict TLS and an ACL on the primary provides all 3 properties for both parties with probably the lowest operational overhead.

8. Policies for Both AXFR and IXFR

We call the entire group of servers involved in XFR (all the primaries and all the secondaries) the 'transfer group'.

Within any transfer group both AXFRs and IXFRs for a zone SHOULD all use the same policy, e.g., if AXFRs use AXoT all IXFRs SHOULD use IXoT.

In order to assure the confidentiality of the zone information, the entire transfer group MUST have a consistent policy of requiring confidentiality. If any do not, this is a weak link for attackers to exploit.

A XoT policy should specify

- If TSIG or SIG(0) is required

- What kind of TLS is required (Opportunistic, Strict or mTLS)

- If IP based ACLs should also be used.

Since this may require configuration of a number of servers who may be under the control of different operators the desired consistency could be hard to enforce and audit in practice.

Certain aspects of the Policies can be relatively easily tested independently, e.g., by requesting zone transfers without TSIG, from unauthorized IP addresses or over cleartext DNS. Other aspects such as if a secondary will accept data without a TSIG digest or if secondaries are using Strict as opposed to Opportunistic TLS are more challenging.

The mechanics of co-ordinating or enforcing such policies are out of the scope of this document but may be the subject of future operational guidance.

9. Implementation Considerations

TBD

10. Implementation Status

The 1.9.2 version of Unbound [8] includes an option to perform AXoT (instead of AXFR-over-TCP). This requires the client (secondary) to authenticate the server (primary) using a configured authentication domain name.

It is noted that use of a TLS proxy in front of the primary server is a simple deployment solution that can enable server side XoT.

11. IANA Considerations

11.1. Registration of XoT Identification String

This document creates a new registration for the identification of XoT in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry [RFC7301].

The "xot" string identifies XoT:

Protocol: XoT

Identification Sequence: 0x64 0x6F 0x72 ("xot")

Specification: This document

12. Security Considerations

This document specifies a security measure against a DNS risk: the risk that an attacker collects entire DNS zones through eavesdropping on clear text DNS zone transfers.

This does not mitigate:

- the risk that some level of zone activity might be inferred by observing zone transfer sizes and timing on encrypted connections (even with padding applied), in combination with obtaining SOA records by directly querying authoritative servers.

- the risk that hidden primaries might be inferred or identified via observation of encrypted connections.

- the risk of zone contents being obtained via zone enumeration techniques.

Security concerns of DoT are outlined in [RFC7858] and [RFC8310].

13. Acknowledgements

The authors thank Benno Overeinder, Shumon Huque and Tim Wicinski for review and discussions.

14. Contributors

Significant contributions to the document were made by:

Han Zhang
Salesforce
San Francisco, CA
United States

Email: hzhang@salesforce.com

15. Changelog

draft-ietf-dprive-xfr-over-tls-02

- Significantly update descriptions for both AXoT and IXoT for message and connection handling taking into account previous specifications in more detail

- Add use of APLN and limitations on traffic on XoT connections.

- Add new discussions of padding for both AXoT and IXoT

- Add text on SIG(0)

- Update security considerations

- Move multi-primary considerations to earlier as they are related to connection handling

draft-ietf-dprive-xfr-over-tls-01

- o Minor editorial updates

- o Add requirement for TLS 1.3. or later

draft-ietf-dprive-xfr-over-tls-00

- o Rename after adoption and reference update.

- o Add placeholder for SIG(0) discussion

- o Update section on ZONEMD

draft-hzpa-dprive-xfr-over-tls-02

- o Substantial re-work of the document.

draft-hzpa-dprive-xfr-over-tls-01

- o Editorial changes, updates to references.

draft-hzpa-dprive-xfr-over-tls-00

- o Initial commit

16. References

16.1. Normative References

[I-D.vcelak-nsec5]
Vcelak, J., Goldberg, S., Papadopoulos, D., Huque, S., and D. Lawrence, "NSEC5, DNSSEC Authenticated Denial of Existence", draft-vcelak-nsec5-08 (work in progress), December 2018.

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

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

[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>.

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

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

[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>.

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

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

16.2. Informative References

[I-D.ietf-dnsop-dns-zone-digest] Wessels, D., Barber, P., Weinberg, M., Kumari, W., and W. Hardaker, "Message Digest for DNS Zones", draft-ietf-dnsop-dns-zone-digest-08 (work in progress), June 2020.

[I-D.ietf-dnsop-extended-error]
Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", draft-ietf-dnsop-extended-error-16 (work in progress), May 2020.

[I-D.ietf-dprive-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnsoquic-00 (work in progress), April 2020.

[I-D.ietf-dprive-phase2-requirements]
Livingood, J., Mayrhofer, A., and B. Overeinder, "DNS Privacy Requirements for Exchanges between Recursive Resolvers and Authoritative Servers", draft-ietf-dprive-phase2-requirements-01 (work in progress), June 2020.

[I-D.vandijk-dprive-ds-dot-signal-and-pin]
Dijk, P., Geuze, R., and E. Bretelle, "Signalling Authoritative DoT support in DS records, with key pinning", draft-vandijk-dprive-ds-dot-signal-and-pin-00 (work in progress), May 2020.

[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>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC2535] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, DOI 10.17487/RFC2535, March 1999, <https://www.rfc-editor.org/info/rfc2535>.

[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>.

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/info/rfc7766>.

16.3. URIs

[1] https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls/blob/master/02-draft-dprive-svg/AXFR_mechanism.svg

[2] https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls/blob/master/02-draft-dprive-svg/IXFR_mechanism.svg

[3] https://www.isc.org/bind/

[4] https://www.nlnetlabs.nl/projects/nsd/about/

[5] https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls/blob/master/02-draft-dprive-svg/AXoT_mechanism.svg

[6] https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls/blob/master/02-draft-dprive-svg/IXoT_mechanism.svg

[7] https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls/blob/02_updates/02-draft-svg/Properties_of_Authentication_methods_for_XoT.svg

[8] https://github.com/NLnetLabs/unbound/blob/release-1.9.2/doc/Changelog

Authors' Addresses

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
The Netherlands

Email: willem@nlnetlabs.nl

Sara Dickinson
Sinodun IT
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: sara@sinodun.com

Shivan Sahib
Salesforce
Vancouver, BC
Canada

Email: ssahib@salesforce.com

Pallavi Aras
Salesforce
Herndon, VA
United States

Email: paras@salesforce.com

Allison Mankin
Salesforce
Herndon, VA
United States

Email: allison.mankin@gmail.com

dprive
Internet-Draft
Intended status: Standards Track
Expires: 14 January 2021

P. van Dijk
PowerDNS
R. Geuze
TransIP
E. Bretelle
Facebook
13 July 2020

Signalling Authoritative DoT support in DS records, with key pinning

draft-vandijk-dprive-ds-dot-signal-and-pin-01

Abstract

This document specifies a way to signal the usage of DoT, and the pinned keys for that DoT usage, in authoritative servers. This signal lives on the parent side of delegations, in DS records. To ensure easy deployment, the signal is defined in terms of (C)DNSKEY.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Even quite recently, DNS was a completely unencrypted protocol, with no protection against snooping. In the past few years, this landscape has shifted. The connections between stubs and resolvers are now often protected by DoT, DoH, or other protocols that provide privacy.

This document introduces a way to signal, from the parent side of a delegation, that the name servers hosting the delegated zone support DoT, and with which TLS/X.509 keys. This proposal does not require any changes in authoritative name servers, other than (possibly through an external process) actually offering DoT on port 853

[RFC7858]. DNS registry operators (such as TLD operators) also need to make no changes, unless they filter uploaded DNSKEY/DS records on acceptable DNSKEY algorithms, in which case they would need to add algorithm TBD to that list.

This document was inspired by, and borrows heavily from, [I-D.bretelle-dprive-dot-for-insecure-delegations].

2. Document work

This document lives on GitHub (https://github.com/PowerDNS/parent-signals-dot/blob/master/draft-vandijk-dprive-ds-dot-signal-and-pin/draft-vandijk-dprive-ds-dot-signal-and-pin.md); proposed text and editorial changes are very much welcomed there, but any functional changes should always first be discussed on the IETF DPRIVE WG (dns-privacy) mailing list.

3. Conventions and Definitions

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.

CDNSKEY record as defined in [RFC7344][RFC8078]

DS record as defined in [RFC4034]

DNSKEY record as defined in [RFC4034]

4. Summary

To enable the signaling of DoT a new DNSKEY algorithm type TBD is added. If a resolver with support for TBD encounters a DS record with the DNSKEY algorithm type TBD it MUST connect to the authoritative servers for this domain via DoT. It MUST use the hashes attached to the DS records with DNSKEY algorithm type TBD to check whether the public key supplied by the authoritative nameserver during the TLS handshake is valid. If the DoT connection is unsuccessful or the public key supplied by the server does not match any of the DS digests, the resolver MUST NOT fall back to unencrypted Do53. For resolvers that are willing to probe for protocol support (DNS over HTTPS, DNS over QUIC), a fallback to other encrypted protocols is allowed if they can satisfy the key pin. This means that if a DS for algo TBD is present, and no name servers satisfy the pin requirement, the response returned to the client is SERVFAIL because no name servers for the domain were available to answer the questions.

A domain MAY have more than one DS record with DNSKEY algorithm TBD. A resolver with support for TBD should then try to verify the public key supplied by the authoritative nameserver against every supplied DS record. Multiple records can be used to support multiple DS digest types, multiple TLS key algorithms, different keys for each authoritative, and for key rollovers. In case of an algorithm or key rollover the new DS record should be added to all served domains before the new key is deployed on the authoritatives. To allow for emergency rollovers, having a standby DS record for all domains with a private key securely stored offline can be a valid strategy.

The pseudo DNSKEY record (when considered in wire format) MUST contain the ([RFC4648] 4.) DER SubjectPublicKeyInfo as defined in [RFC5280] 4.1.2.7. Since the cert provided by the TLS server over the wire is already DER encoded this makes for easy validation. (In the DNSKEY presentation format, the Public Key field contains the Base64 encoding of the DER SPKI, which is equivalent to the SPKI in PEM format minus the header and footer.) The pseudo DNSKEY algorithm type TBD is algorithm agnostic, like the TLSA record, since the DER encoded data already contains information about the used algorithm. Algorithm support SHOULD be handled at the TLS handshake level, which means a DNS application SHOULD NOT need to be aware of the algorithm used by its TLS library. The pseudo DNSKEY record MUST NOT be present in the zone. The procedure for hashing the pseudo DNSKEY record is the same as for a normal DNSKEY as defined in RFC4034 5.1.4.

As DNSKEY algorithm TBD is not meant to be used for Zone Signing, the existing ZONE and SEP flags do not mean anything. This specification statically defines the flags value as 257 for optimal compatibility with existing registry operations.

The pseudo DNSKEY type can be used in CDNSKEY and CDS (as defined in [RFC7344]) records. These records MAY be present in the zone.

For those familiar with TLSA ([RFC6698]), key matching for this protocol is identical to that provided by "TLSA 3 1 0" for (C)DNSKEY. For the DS case, key matching is similar to "TLSA 3 1 x" where x is not zero, except that the rest of the (C)DNSKEY, including the owner name, gets prepended before hashing.

5. Example

This section will take you through the various parts of this specification, by example.

We assume that we are working with a domain "example.com." with one name server, "ns.example.com.".

5.1. Generating and placing the (C)DNSKEY/DS records

[NOTE: this section uses '225' instead of 'TBD' because otherwise the code does not work. We need to fix this before publication.]

We will walk you through the CDNSKEY/DS generation, demonstrating it in terms of basic shell scripting and some common tools.

First, we extract the SubjectPublicKeyInfo:

```
openssl s_client -connect ns.example.com:853 < /dev/null \
  | openssl x509 -noout -pubkey > pubkey.pem
```

This gives us a file "pubkey.pem" that looks like this (abridged):

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAxH2a6NxIcw5527b04kKy
...
71AWASNoX2GQh7eaQPDD9i8CAwEAAQ==
-----END PUBLIC KEY-----
```

To turns this into a CDNSKEY:

1. remove the header and footer

2. remove all newlines

In other words:

```
openssl s_client -connect ns.example.com:853 </dev/null \
  | openssl x509 -noout -pubkey \
  | sed '1d;$d' \
  | tr -d '\n'
```

Then we prepend

```
example.com. IN CDNSKEY 257 3 225
```

so that we end up with

```
example.com. IN CDNSKEY 257 3 225 MIICIj...AAQ==
```

If your registry accepts CDNSKEY, or DNSKEY via EPP, you are done - you can get your DS placed.

To generate the DS, do something like this:

```
echo example.com. IN DNSKEY 257 3 225 MIICIj...AAQ== \
  | ldns-key2ds -f -n -2 /dev/stdin
example.com.    3600    IN      DS      7573 225 2 fcb6...c26c
```

6. Implementation

The subsection titles in this section attempt to follow the terminology from [RFC8499] in as far as it has suitable terms. 'Implementation' is understood to mean both 'code changes' and 'operational changes' here.

6.1. Authoritative server changes

This specification defines no changes to query processing in authoritative servers.

If DoT-signaling DS records are published for a zone, all name servers for the zone (from both the parent-side and child-side NS RRsets) SHOULD offer DoT service on port 853, and when they do, they SHOULD do so using keys present in the DS RRset. However, there are potential cases where this is not possible, like having multiple DNS providers. In this case the name servers that do not support DoT MUST respond with a RST response or similar on the port tcp/853 to prevent name resolution slowdowns.

6.2. Validating resolver changes

If a resolver successfully uses DoT with a nameserver as specified in this document for one domain, it MAY assume DoT is always available from that nameserver for questions for another domain. However, it MUST NOT assume that the connection is properly pinned for that other domain unless there is a DS record available for that other domain it is currently resolving.

A validating resolver that supports this draft will perform the following actions when a DS record with algorithm TBD is encountered:

1. Connects to the name server on port 853.

2. During TLS handshake, the resolver will extract the SubjectPublicKeyInfo from the certificate.

3. Construct an in-memory DNSKEY record [RFC4034] section 2 with its fields set as follow:

   * Flags: 257

   * Protocol: 3

   * Algorithm: TBD

   * Public Key: The wire-format SubjectPublicKeyInfo

4. Get the list of Digest Type for DS records obtained from the parent with algorithm TBD

5. For each digest type from the list, compute the DS record of the previously computed DNSKEY, its fields are set as follow:

   * Key Tag: computed from DNS key using [RFC4034] appendix B

   * Algorithm: TBD

   * Digest Type: the current Digest Type we are computing the DS for.

   * Digest: Following [RFC4034] section 5.1.4, compute the digest of owner name | previously computed DNSKEY's RDATA.

6. Test the computed DS record against all the supplied DS records until a match is encountered.

7. If any computed DS record matches a DS record in the DS record set we got from the parent, the connection is successfully authenticated.

6.3. Stub resolver changes

This specification defines no changes to stub resolvers.

6.4. Zone validator changes

This section covers both the 'online' type of zone validator, such as Zonemaster, and the 'offline full zone' type, such as "validns" and "dnssec-verify".

Checks for child DNSKEY records based on parent DS records algorithms, and checks for zone RRSIG algorithms based on DNSKEY algorithms, MUST not be applied to algorithm TBD. [NOTE: rephrase this in terms of the Zone Signing column at https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml (https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml) ?]

DNSKEY validity checks MAY verify correct DER syntax in DNSKEY Public Key content when algorithm is TBD.

6.5. Domain registry changes

Any pre-delegation or periodic checks by registries should honor the Zone validator changes from the previous section.

This specification trusts that appearance of TBD in https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml (https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml) will eventually lead registries to accept DS/(C)DNSKEY submissions for algorithm TBD.

Registries that limit the total number of DS records for a delegation SHOULD consider having a separate limit for algorithm TBD DS records, as their management is separate from actual DNSSEC key management.

7. Security Considerations

This document defines a way to convey, authoritatively, that resolvers must use DoT to do their queries to the name servers for a certain zone. By doing so, that exchange gains confidentiality, data integrity, peer entity authentication.

8. Implementation Status

[RFC Editor: please remove this section before publication]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this document, and is based on a proposal described in [RFC6982]. 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 RFC 6982, "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".

8.1. PoC

Some Proof of Concept code showing the generation of the (C)DNSKEY, and the subsequent hashing by a client (which should match one of the DS records with algo TBD), in Python and Go, is available at https://github.com/PowerDNS/parent-signals-dot/tree/master/poc (https://github.com/PowerDNS/parent-signals-dot/tree/master/poc)

9. Design Considerations

[RFC Editor: please remove this section before publication]

A protocol design is nothing without a clear statement of the constraints it was designed to meet, and perhaps a list of other constraints it meets by accident.

We humbly acknowledge Petr Spacek's excellent summary of the 'nice properties' this protocol has (https://mailarchive.ietf.org/arch/msg/dns-privacy/_Zf5TGVAcUfPRrQ_7o_NPnmnlZs/) as a source of inspiration for this section.

Manu's DSPKI proposal had the following excellent properties:

* no extra roundtrips (assuming DSPKI came 'for free' with delegations like DS records do today)

* downgrade resistance

* simple protocol, no indirections

It also had this one very important undesirable property:

* a new RRtype with 'special' behaviour would be pretty much impossible to deploy

In various private and public discussions, it was quickly realised that fitting this into the actual DS record would solve that problem. The first obvious answer to that is 'just assign some numbers and do in DS what DSPKI defined in its own type'. Petr Spacek and others pointed out that this would be incompatible with 'DNSKEY-style' registries, i.e. those that demand DNSKEY, not DS, in their communications (those communications being either EPP, some registry-specific protocol, or CDNSKEY). In other words, a protocol that would not allow the DS to be hashed 'the usual way' from a DNSKEY would not go far, as many registries are slow to update their software even _just_ for a couple of new numbers in an IANA registry.

With that, the puzzle was clear. We need some format to signal and pin DoT with a DNSKEY, in such a way that a DS can be hashed from it without software changes in parties such as registries, and such that that DS is enough for a resolver to validate a TLS connection.

Eventually we realised that a resolver could take the TLS SubjectPublicKeyInfo, construct a 'pseudo' DNSKEY from it, and hash that into a DS. This resolves the one bad property of DSPKI (deployability without changing every auth, resolver, and registry stack in the world).

The design constraints we felt we must meet with this protocol were:

* deployability without demanding massive software changes or even 'flag days'

* downgrade resistance

And we feel we have met those. The other positive properties of DSPKI (simplicity, no extra roundtrips) have been kept intact more by accident than by strong intention.

We can understand that several people are saying that this is hacky (we do not even disagree), and that TLSA should have been used. However, we feel that any TLSA-based protocol we can imagine would be a lot more complex, and therefore prone to breakage which might be hard to debug. It would also be very easy to accidentally introduce chicken-and-egg problems with a more indirect approach. Note that we are responding to imagined TLSA-based protocols here. If a draft appears for a TLSA-based approach to DoT signaling/pinning, we would love to read it. Depending on what that draft looks like, it might even make sense to have that protocol _and_ the protocol described in this document.

The biggest downside to this DS-based protocol is that a change in TLS keys on an auth may require DS updates for thousands or even hundreds of thousands of domains. This issue is partially mitigated by allowing backup keys to be part of those DS sets. Furthermore we hope that efforts from Cloudflare and others for shortening the path between auth operator and domain registrar one day work out. Those efforts are focused on NSset updates and DS updates for DNSSEC validation, but they would also aid key rollovers for this protocol greatly.

10. IANA Considerations

This document updates the IANA registry "DNS Security Algorithm Numbers" at https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml (https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml)

The following entries have been added to the registry:

Number	TBD
Description	DoT signal+pin
Mnemonic	DOTPIN
Zone signing	N
Trans sec.	N
Reference	RFC TBD2

11. Acknowledgements

The authors would like to thank the following individuals for their useful input: Job Snijders, Maik Zumstrull, Petr Spacek, Pieter Lexis, Ralph Dolmans, Remi Gacogne, Seth Arnold, and Vladimir Cunat.

12. Normative References

[I-D.bretelle-dprive-dot-for-insecure-delegations]
Bretelle, E., "DNS-over-TLS for insecure delegations", Work in Progress, Internet-Draft, draft-bretelle-dprive-dot-for-insecure-delegations-01, 11 March 2019, <https://tools.ietf.org/html/draft-bretelle-dprive-dot-for-insecure-delegations-01>.

[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>.

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

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[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>.

[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>.

[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>.

[RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <https://www.rfc-editor.org/info/rfc8078>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

13. Informative References

[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>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

Appendix A. Document history

A.1. Changes between -00 and -01

1. Lots of clarifying text that does not change any semantics, including:

   * a section on how resolvers would actually use this protocol.

   * we made it clearer that multiple DS records for a delegation are allowed, and why you would want this.

2. DNSKEY flags are now set to 257, because it looks like this will make it a lot easier for many registries to accept the records.

3. Added a 'Design Considerations' section to give some background to why this protocol is what it is.

We have tried to do a review of this protocol against the requirement of the DPRIVE phase 2 document. You can find this review (which might be updated outside of revisions of this draft or the phase 2 draft) in our GitHub repo (https://github.com/PowerDNS/parent-signals-dot/blob/master/draft-vandijk-dprive-ds-dot-signal-and-pin/yardsticks/draft-ietf-dprive-phase2-requirements-01.md).

Authors' Addresses

Peter van Dijk
PowerDNS
Den Haag
Netherlands

Email: peter.van.dijk@powerdns.com

Robin Geuze
TransIP
Delft
Netherlands

Email: robing@transip.nl

Emmanuel Bretelle
Facebook

Email: chantra@fb.com