      Multi-homing Deployment Considerations for Distributed-Denial-of-Service
                    Open Threat Signaling (DOTS)
                 draft-boucadair-dots-multihoming-02

Abstract

   This document discusses multi-homing considerations for Distributed-
   Denial-of-Service Open Threat Signaling (DOTS). The goal is to
   provide a set of guidance for DOTS clients/gateways when multihomed.

Requirements Language

   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 RFC
   2119 [RFC2119].

Table of Contents

1. Introduction

   In many deployments, it may not be possible for a network to
   determine the cause for a distributed Denial-of-Service (DoS) attack
   [RFC4732], but instead just realize that some resources seem to be
   under attack. To fill that gap, the IETF is specifying an
   architecture, called DDoS Open Threat Signaling (DOTS)
   [I-D.ietf-dots-architecture], in which a DOTS client can inform a
   DOTS server that the network is under a potential attack and that
   appropriate mitigation actions are required. Indeed, because the
   lack of a common method to coordinate a real-time response among
   involved actors and network domains inhibits the effectiveness of
   DDoS attack mitigation, DOTS protocol is meant to carry requests for
   DDoS attack mitigation, thereby reducing the impact of an attack and

leading to more efficient defensive actions.
[I-D.ietf-dots-use-cases] identifies a set of scenarios for DOTS;
almost all these scenarios involve a CPE.

The basic high-level DOTS architecture is illustrated in Figure 1
([I-D.ietf-dots-architecture]):

```
    +-----------+                +-------------+
    | Mitigator | ~~~~~~~~~~ | DOTS Server |
    +-----------+                +-------------+
                                       |
                                       |
                                       |
                                       |
    +---------------+         +-------------+
    | Attack Target | ~~~~~ | DOTS Client |
    +---------------+         +-------------+
```

Figure 1: Basic DOTS Architecture

[I-D.ietf-dots-architecture] specifies that the DOTS client may be
provided with a list of DOTS servers; each associated with one or
more IP addresses. These addresses may or may not be of the same
address family. The DOTS client establishes one or more DOTS
signaling sessions by connecting to the provided DOTS server(s)
addresses.

DOTS may be deployed within networks that are connected to one single
upstream provider. It can also be enabled within networks that are
multi-homed. The reader may refer to [RFC3582] for an overview of
multi-homing goals and motivations. This document discusses DOTS
multi-homing considerations. Specifically, the document aims to:

1. Complete the base DOTS architecture with multi-homing specifics.
    Those specifics need to be taking into account because:

    *   Send a DOTS mitigation request to an arbitrary DOTS server
        won't help mitigating a DDoS attack.

    *   Blindly forking all DOTS mitigation requests among all
        available DOTS servers is suboptimal.

    *   Sequentially contacting DOTS servers may increase the delay
        before a mitigation plan is enforced.

2. Identify DOTS deployment schemes in a multi-homing context, where
    DOTS service can be offered by all or a subset of upstream
    providers.

3. Sketch guidelines and recommendations for placing DOTS requests
    in multi-homed networks, e.g.,:

    * Select the appropriate DOTS server(s).

    * Identify cases where anycast is not recommended.

   To that aim, this document adopts the following methodology:

   o Identify and extract viable deployment candidates from
      [I-D.ietf-dots-use-cases].

   o Augment the description with multi-homing technicalities, e.g.,

      * One vs. multiple upstream network providers

      * One vs. multiple interconnect routers

      * Provider-Independent (PI) vs. Provider-Aggregatable (PA)

   o Describe the recommended behavior of DOTS clients and gateways for
      each case.

   Multi-homed DOTS agents are assumed to make use of the protocols
   defined in [I-D.ietf-dots-signal-channel] and
   [I-D.ietf-dots-data-channel]; no specific extension is required to
   the base DOTS protocols for deploying DOTS in a multihomed context.

2. Terminology

   This document makes use of the terms defined in
   [I-D.ietf-dots-architecture] and [RFC4116].

   IP refers to both IPv4 and IPv6.

3. Multi-Homing Scenarios

   This section briefly describes some multi-homing scenarios that are
   relevant to DOTS. In the following sub-sections, only the
   connections of border routers are shown; internal network topologies
   are not elaborated hereafter.

3.1. Residential CPE

   The scenario shown in Figure 2 is characterized as follows:

   o The home network is connected to the Internet using one single CPE
      (Customer Premises Equipment).

o The CPE is connected to multiple provisioning domains (i.e. both
   fixed and mobile networks). Provisioning domain (PvD) is
   explained in [RFC7556].

o Each of these provisioning domains assign IP addresses/prefixes to
   the CPE. These addresses/prefixes are said to be Provider-
   Aggregatable (PA).

o The CPE is provided by each of these provisioning domains with
   additional configuration information such as a list of DNS
   servers, DNS suffixes associated with the network, default gateway
   address, and DOTS server's name
   [I-D.boucadair-dots-server-discovery].

o Because of ingress filtering, packets forwarded by the CPE to a
   given provisioning domain must be send with a source IP address
   that was assigned by that network [RFC8043].

```
          +-------+              +-------+
          |Fixed  |              |Mobile |
          |Network|              |Network|
          +---+---+              +---+---+
              |                      |      Service Providers
      ........|......................|........................
          +---------++---------+         Home Network
                    ||
              +--++-+
              | CPE |
              +-----+
                    ... (Internal Network)
```

Figure 2: Typical Multi-homed Residential CPE

3.2. Multi-homed Enterprise: Single CPE, Multiple Upstream ISPs

   The scenario shown in Figure 3 is characterized as follows:

   o The enterprise network is connected to the Internet using one
      single router.

   o That router is connected to multiple provisioning domains (i.e.
      managed by distinct administrative entities).

   Unlike the previous scenario, two sub-cases can be considered for an
   enterprise network with regards to assigned addresses:

1. Provider Independent (PI) addresses: The enterprise is the owner
    of the IP addresses/prefixes; the same address/prefix is then
    used for communication placed using any of the provisioning
    domains.

2. PA addresses/prefixes: each of provisioning domains assigns IP
    addresses/prefixes to the enterprise network.

```
         +------+                  +------+
         | ISP1 |                  | ISP2 |
         +---+--+                  +--+---+
             |                        |      Service Providers
.............|........................|.......................
            +---------++---------+       Enterprise Network
                      ||
                    +--++-+
                    | rtr |
                    +-----+
                       ... (Internal Network)
```

Figure 3: Multi-homed Enterprise Network (Single CPE connected to
Multiple Networks)

3.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

This scenario is similar to the one in Section 3.2; the main
difference is that dedicated routers are used to connect to each
provisioning domain.

```
         +------+    +------+
         | ISP1 |    | ISP2 |
         +---+--+    +--+---+
             |          |        Service Providers
.....................|..........|.......................
             |          |        Enterprise Network
         +---+--+    +--+---+
         | rtr1 |    | rtr2 |
         +------+    +------+

             ... (Internal Network)
```

Figure 4: Multi-homed Enterprise Network (Multiple CPEs, Multiple
ISPs)

3.4. Multi-homed Enterprise with the Same ISP

   This scenario is a variant of Section 3.2 and Section 3.3 in which
   multi-homing is provided by the same ISP (i.e., same provisioning
   domain).

4. DOTS Deployment Considerations

   Table 1 provides some sample (non-exhaustive) deployment schemes to
   illustrate how DOTS agents may be deployed for each of the scenarios
   introduced in Section 3.

   +--------------------------+------------------------+------------+
   | Scenario | DOTS client | DOTS |
   | | | gateway |
   +--------------------------+------------------------+------------+
   | Residential CPE | CPE | N/A |
   +--------------------------+------------------------+------------+
   | Single CPE, Multiple | internal hosts or CPE | CPE |
   | provisioning domains | | |
   +--------------------------+------------------------+------------+
   | Multiple CPEs, Multiple| internal hosts or all | CPEs (rtr1 |
   | provisioning domains | CPEs (rtr1 and rtr2) | and rtr2) |
   +--------------------------+------------------------+------------+
Multi-homed enterprise,	internal hosts or all	CPEs (rtr1
Single provisioning	CPEs (rtr1 and rtr2)	and rtr2)
domain		
   +--------------------------+------------------------+------------+

                      Table 1: Sample Deployment Cases

   These deployment schemes are further discussed in the following sub-
   sections.

4.1. Residential CPE

   Figure 5 depicts DOTS signaling sessions that are required to be
   established between a DOTS client (C) and DOTS servers (S1, S2) in
   the context of the scenario described in Section 3.1.

   The DOTS client MUST resolve the DOTS server's name provided by a
   provisioning domain ([I-D.boucadair-dots-server-discovery]) using the
   DNS servers learned from the same provisioning domain. The DOTS
   client MUST use the source address selection algorithm defined in
   [RFC6724] to select the candidate source addresses to contact each of
   these DOTS servers. DOTS signaling sessions must be established and
   maintained with each of the DOTS servers because the mitigation scope
   of these servers is restricted. The DOTS client SHOULD use the

certificate provisioned by a provisioning domain to authenticate
itself to the DOTS server provided by the same provisioning domain.
When conveying a mitigation request to protect the attack target(s),
the DOTS client among the DOTS servers available MUST select a DOTS
server whose network has assigned the prefixes from which target
prefixes and target IP addresses are derived. For example,
mitigation request to protect target resources bound to a PA IP
address/prefix cannot be honored by an provisioning domain other than
the one that owns those addresses/prefixes. Consequently, Typically,
if a CPE detects a DDoS attack on all its network attachments, it
must contact both DOTS servers for mitigation. Nevertheless, if the
DDoS attack is received from one single network, then only the DOTS
server of that network must be contacted.

The DOTS client MUST be able to associate a DOTS server with each
provisioning domain. For example, if the DOTS client is provisioned
with S1 using DHCP when attaching to a first network and with S2
using Protocol Configuration Option (PCO) when attaching to a second
network, the DOTS client must record the interface from which a DOTS
server was provisioned. DOTS signaling session to a given DOTS
server must be established using the interface from which the DOTS
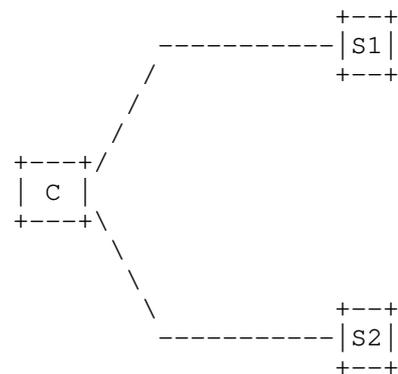server was provisioned.

```
                                      +--+
                          ----------|S1|
                         /            +--+
                        /
                       /
                      /
           +---+/
           | C |
           +---+\
                 \
                  \
                   \            +--+
                    ----------|S2|
                                +--+
```

        Figure 5: DOTS associations for a multihomed residential CPE

4.2. Multi-homed Enterprise: Single CPE, Multiple Upstream ISPs

   Figure 6 illustrates a first set of DOTS associations that can be
   established with a DOTS gateway is enabled in the context of the
   scenario described in Section 3.2. This deployment is characterized
   as follows:

   o One of more DOTS clients are enabled in hosts located in the
      internal network.

   o A DOTS getaway is enabled to aggregate/relay the requests to
      upstream DOTS servers.

   When PA addresses/prefixes are in used, the same considerations
   discussed in Section 4.1 are to be followed by the DOTS gateway to
   contact its DOTS server(s). The DOTS gateways can be reachable from
   DOTS client using a unicast or anycast address.

   Nevertheless, when PI addresses/prefixes are assigned, the DOTS
   gateway MUST sent the same request to all its DOTS servers.

```
                                         +--+
                              ----------|S1|
            +---+                /        +--+
            | C1|----+     /
            +---+    |   /
       +---+      +-+-+/
       | C3|------| G |
       +---+      +-+-+\
         +---+     |   \
         | C2|----+     \
         +--+           \       +--+
                         ----------|S2|
                                   +-+
```
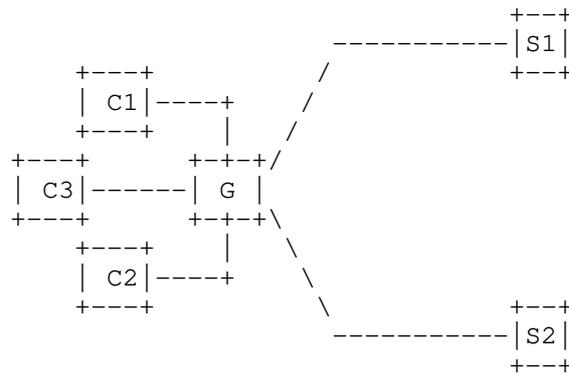
         Figure 6: Multiple DOTS Clients, Single DOTS Gateway, Multiple DOTS
                                  Servers

   An alternate deployment model is depicted in Figure 7. This
   deployment assumes that:

   o One of more DOTS clients are enabled in hosts located in the
      internal network. These DOTS client may use
      [I-D.boucadair-dots-server-discovery] to discover its DOTS
      server(s).

   o These DOTS clients communicate directly with upstream DOTS
      servers.

   If PI addresses/prefixes are in use, the DOTS client can send the
   mitigation request for all its PI addresses/prefixes to any one of
   the DOTS servers. The use of anycast addresses is NOT RECOMMENDED.

   If PA addresses/prefxies are used, the same considerations discussed
   in Section 4.1 are to be followed by the DOTS clients. Because DOTS
   clients are not located on the CPE and multiple addreses/prefixes may
   not be assigned to the DOTS client (IPv4 context, typically), some
   complications arise to steer the traffic to the appropriate DOTS

server using the appropriate source IP address. These complications
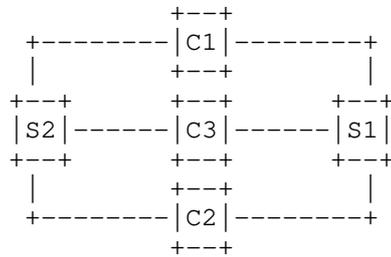discussed in [RFC4116] are not specific to DOTS .

```
                      +--+
          +--------|C1|--------+
          |          +--+          |
  +--+        +--+        +--+
  |S2|------|C3|------|S1|
  +--+        +--+        +--+
          |          +--+          |
          +--------|C2|--------+
                      +--+
```

Figure 7: Multiple DOTS Clients, Multiple DOTS Servers

Another deployment approach is to enable many DOTS clients; each of
them responsible to handle communication with a specific DOTS server
(see Figure 8). Each DOTS client is provided with policies (e.g.,
prefix filter) that will trigger DOTS communications with the DOTS
servers. The CPE MUST select the appropriate source IP address when
forwarding DOTS messages received from an internal DOTS client. If
anycast addresses are used to reach DOTS servers, the CPE may not be
able to select the appropriate provisioning domain to which the
mitigation request should be forwarded. As a consequence, the
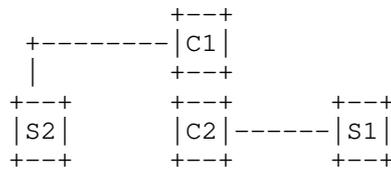request may not be forwarded to the appropriate DOTS server.

```
                      +--+
          +--------|C1|
          |          +--+
  +--+        +--+        +--+
  |S2|        |C2|------|S1|
  +--+        +--+        +--+
```

Figure 8: Single Homed DOTS Clients

4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

The deployments depicted in Figure 7 and Figure 8 apply also for the
scenario described in Section 3.3. One specific problem for this
scenario is to select the appropriate exit router when contacting a
given DOTS server.

An alternative deployment scheme is shown in Figure 9:

o DOTS clients are enabled in hosts located in the internal network.

o A DOTS gateway is enabled in each CPE (rtr1, rtr2).

   o Each of these DOTS gateways communicate with the DOTS server of
      the provisioning domain.

   When PI addresses/prefixes are used, DOTS clients can contact any of
   the DOTS gateways to send a DOTS message. DOTS gateway will then
   relay the request to the DOTS server. Note that the use of anycast
   addresses is NOT RECOMMENDED to establish DOTS signaling sessions
   between DOTS client and DOTS gateways.

   When PA addresses/prefixes are used, but no filter rules are provided
   to DOTS clients, these later MUST contact all DOTS gateways
   simultaneously to send a DOTS message. Upon receipt of a request by
   a DOTS gateway, it MUST check whether the request is to be forwarded
   upstream or be rejected.

   When PA addresses/prefixes are used, but specific filter rules are
   provided to DOTS clients using some means that are out of scope,
   these later MUST select the appropriate DOTS gateway to be contacted.
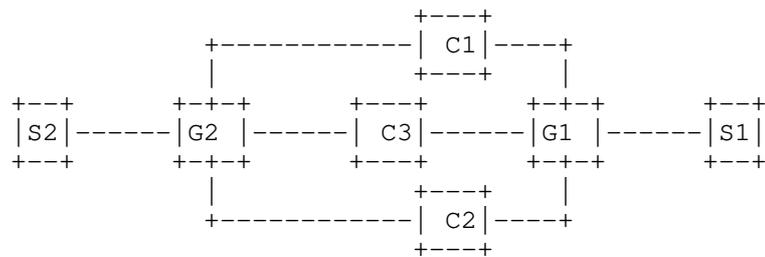   The use of anycast is NOT RECOMMENDED to reach DOTS gateways.

```
                                    +---+
                      +-----------| C1|----+
                      |             +---+    |
    +--+         +-+-+         +---+         +-+-+         +--+
    |S2|------|G2 |------| C3|------|G1 |------|S1|
    +--+         +-+-+         +---+         +-+-+         +--+
                      |             +---+    |
                      +-----------| C2|----+
                                    +---+
```

            Figure 9: Multiple DOTS Clients, Multiple DOTS Gateways, Multiple
                               DOTS Servers

4.4. Multi-homed Enterprise: Single ISP

   The key difference of the scenario described in Section 3.4 compared
   to the other scenarios is that multi-homing is provided by the same
   ISP. Concretely, that ISP can decided to provision the enterprise
   network with:

   1. The same DOTS server for all network attachments.

   2. Distinct DOTS servers for each network attachment. These DOTS
       servers needs to coordinate when a mitigation action is received
       from the enterprise network.

In both cases, DOTS agents enabled within the enterprise network may
decide to select one or all network attachments to place DOTS
mitigation requests.

5. Security Considerations

   DOTS-related security considerations are discussed in Section 4 of
   [I-D.ietf-dots-architecture].

   TBD: In Home networks, if EST is used then how will the DOTS gateway
   (EST client) be provisioned with credentials for initial enrolment
   (see Section 2.2 in RFC 7030).

6. IANA Considerations

   This document does not require any action from IANA.

7. Acknowledgements

   Thanks to Roland Dobbins and Nik Teague for sharing their comments on
   the mailing list.

   Thanks to Kirill Kasavchenko for the comments.

8. References

8.1. Normative References

   [I-D.ietf-dots-architecture]
             Mortensen, A., Andreasen, F., Reddy, T.,
             christopher_gray3@cable.comcast.com, c., Compton, R., and
             N. Teague, "Distributed-Denial-of-Service Open Threat
             Signaling (DOTS) Architecture", draft-ietf-dots-
             architecture-04 (work in progress), July 2017.

   [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>.

   [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
             "Default Address Selection for Internet Protocol Version 6
             (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
             <https://www.rfc-editor.org/info/rfc6724>.

8.2. Informative References

   [I-D.boucadair-dots-server-discovery]
              Boucadair, M., Reddy, T., and P. Patil, "Distributed-
              Denial-of-Service Open Threat Signaling (DOTS) Server
              Discovery", draft-boucadair-dots-server-discovery-02 (work
              in progress), July 2017.

   [I-D.ietf-dots-data-channel]
              Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., Patil,
              P., Mortensen, A., and N. Teague, "Distributed Denial-of-
              Service Open Threat Signaling (DOTS) Data Channel", draft-
              ietf-dots-data-channel-05 (work in progress), October
              2017.

   [I-D.ietf-dots-signal-channel]
              Reddy, T., Boucadair, M., Patil, P., Mortensen, A., and N.
              Teague, "Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-
              channel-05 (work in progress), October 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-07 (work
              in progress), July 2017.

   [RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-
              Multihoming Architectures", RFC 3582,
              DOI 10.17487/RFC3582, August 2003,
              <https://www.rfc-editor.org/info/rfc3582>.

   [RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V.
              Gill, "IPv4 Multihoming Practices and Limitations",
              RFC 4116, DOI 10.17487/RFC4116, July 2005,
              <https://www.rfc-editor.org/info/rfc4116>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain
              Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015,
              <https://www.rfc-editor.org/info/rfc7556>.

   [RFC8043] Sarikaya, B. and M. Boucadair, "Source-Address-Dependent
              Routing and Source Address Selection for IPv6 Hosts:
              Overview of the Problem Space", RFC 8043,
              DOI 10.17487/RFC8043, January 2017,
              <https://www.rfc-editor.org/info/rfc8043>.

Authors' Addresses

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

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

   Email: TirumaleswarReddy_Konda@McAfee.com

    Multi-homing Deployment Considerations for Distributed-Denial-of-Service
                     Open Threat Signaling (DOTS)
                 draft-boucadair-dots-multihoming-04

   Abstract

      This document discusses multi-homing considerations for Distributed-
      Denial-of-Service Open Threat Signaling (DOTS). The goal is to
      provide a set of guidance for DOTS clients/gateways when multihomed.

Table of Contents

1. Introduction

   In many deployments, it may not be possible for a network to
   determine the cause for a distributed Denial-of-Service (DoS) attack
   [RFC4732], but instead just realize that some resources seem to be
   under attack. To fill that gap, the IETF is specifying an
   architecture, called DDoS Open Threat Signaling (DOTS)
   [I-D.ietf-dots-architecture], in which a DOTS client can inform a
   DOTS server that the network is under a potential attack and that
   appropriate mitigation actions are required. Indeed, because the
   lack of a common method to coordinate a real-time response among
   involved actors and network domains inhibits the effectiveness of
   DDoS attack mitigation, DOTS protocol is meant to carry requests for
   DDoS attack mitigation, thereby reducing the impact of an attack and
   leading to more efficient defensive actions.
   [I-D.ietf-dots-use-cases] identifies a set of scenarios for DOTS;
   almost all these scenarios involve a CPE.

   The basic high-level DOTS architecture is illustrated in Figure 1
   ([I-D.ietf-dots-architecture]):

```
+-----------+                +-------------+
| Mitigator | ~~~~~~~~~~ | DOTS Server |
+-----------+                +-------------+
                                    |
                                    |
                                    |
+----------------+             +-------------+
| Attack Target  | ~~~~~ | DOTS Client |
+----------------+             +-------------+
```
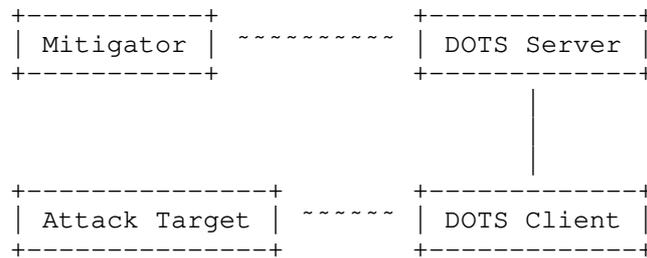
Figure 1: Basic DOTS Architecture

[I-D.ietf-dots-architecture] specifies that the DOTS client may be
provided with a list of DOTS servers; each associated with one or
more IP addresses. These addresses may or may not be of the same
address family. The DOTS client establishes one or more DOTS
signaling sessions by connecting to the provided DOTS server(s)
addresses.

DOTS may be deployed within networks that are connected to one single
upstream provider. It can also be enabled within networks that are
multi-homed. The reader may refer to [RFC3582] for an overview of
multi-homing goals and motivations. This document discusses DOTS
multi-homing considerations. Specifically, the document aims to:

1. Complete the base DOTS architecture with multi-homing specifics.
    Those specifics need to be taking into account because:

    * Send a DOTS mitigation request to an arbitrary DOTS server
       won't help mitigating a DDoS attack.

    * Blindly forking all DOTS mitigation requests among all
       available DOTS servers is suboptimal.

    * Sequentially contacting DOTS servers may increase the delay
       before a mitigation plan is enforced.

2. Identify DOTS deployment schemes in a multi-homing context, where
    DOTS service can be offered by all or a subset of upstream
    providers.

3. Sketch guidelines and recommendations for placing DOTS requests
    in multi-homed networks, e.g.,:

    * Select the appropriate DOTS server(s).

    * Identify cases where anycast is not recommended.

To that aim, this document adopts the following methodology:

o Identify and extract viable deployment candidates from
   [I-D.ietf-dots-use-cases].

o Augment the description with multi-homing technicalities, e.g.,

   * One vs. multiple upstream network providers

   * One vs. multiple interconnect routers

   * Provider-Independent (PI) vs. Provider-Aggregatable (PA)

o Describe the recommended behavior of DOTS clients and gateways for
   each case.

Multi-homed DOTS agents are assumed to make use of the protocols
defined in [I-D.ietf-dots-signal-channel] and
[I-D.ietf-dots-data-channel]; no specific extension is required to
the base DOTS protocols for deploying DOTS in a multihomed context.

2. Requirements Language

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. Terminology

This document makes use of the terms defined in
[I-D.ietf-dots-architecture] and [RFC4116].

IP refers to both IPv4 and IPv6.

4. Multi-Homing Scenarios

This section briefly describes some multi-homing scenarios that are
relevant to DOTS. In the following sub-sections, only the
connections of border routers are shown; internal network topologies
are not elaborated hereafter.

4.1. Residential CPE

The scenario shown in Figure 2 is characterized as follows:

o The home network is connected to the Internet using one single CPE
   (Customer Premises Equipment).

o The CPE is connected to multiple provisioning domains (i.e. both
   fixed and mobile networks). Provisioning domain (PvD) is
   explained in [RFC7556].

o Each of these provisioning domains assign IP addresses/prefixes to
   the CPE. These addresses/prefixes are said to be Provider-
   Aggregatable (PA).

o The CPE is provided by each of these provisioning domains with
   additional configuration information such as a list of DNS
   servers, DNS suffixes associated with the network, default gateway
   address, and DOTS server's name
   [I-D.boucadair-dots-server-discovery].

o Because of ingress filtering, packets forwarded by the CPE to a
   given provisioning domain must be send with a source IP address
   that was assigned by that network [RFC8043].

```
            +-------+               +-------+
            |Fixed  |               |Mobile |
            |Network|               |Network|
            +---+---+               +---+---+
                |                       |     Service Providers
        .............|.....................|........................
               +---------++---------+    Home Network
                        ||
                  +--++-+
                  | CPE |
                  +-----+
                      ... (Internal Network)
```

Figure 2: Typical Multi-homed Residential CPE

4.2. Multi-homed Enterprise: Single CPE, Multiple Upstream ISPs

   The scenario shown in Figure 3 is characterized as follows:

   o The enterprise network is connected to the Internet using one
      single router.

   o That router is connected to multiple provisioning domains (i.e.
      managed by distinct administrative entities).

Unlike the previous scenario, two sub-cases can be considered for an
enterprise network with regards to assigned addresses:

1. Provider Independent (PI) addresses: The enterprise is the owner
    of the IP addresses/prefixes; the same address/prefix is then
    used for communication placed using any of the provisioning
    domains.

2. PA addresses/prefixes: each of provisioning domains assigns IP
    addresses/prefixes to the enterprise network.

```
                +------+              +------+
                | ISP1 |              | ISP2 |
                +---+--+              +--+---+
                    |                    |     Service Providers
         ...........|....................|.......................
                +---------++---------+      Enterprise Network
                         ||
                     +--++-+
                     | rtr |
                     +-----+
                        ... (Internal Network)
```
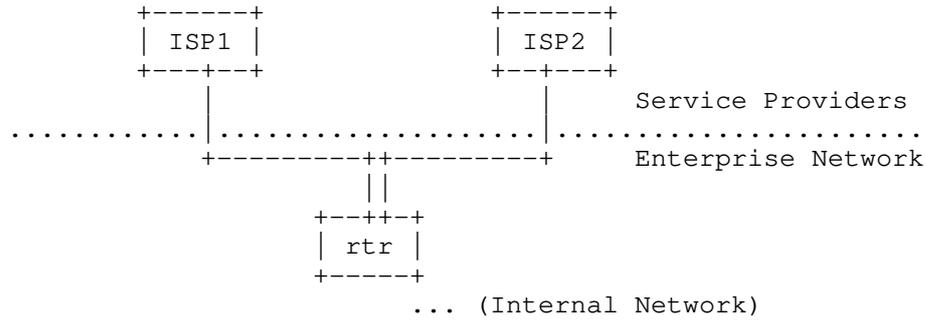
          Figure 3: Multi-homed Enterprise Network (Single CPE connected to
                          Multiple Networks)

4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

   This scenario is similar to the one in Section 4.2; the main
   difference is that dedicated routers are used to connect to each
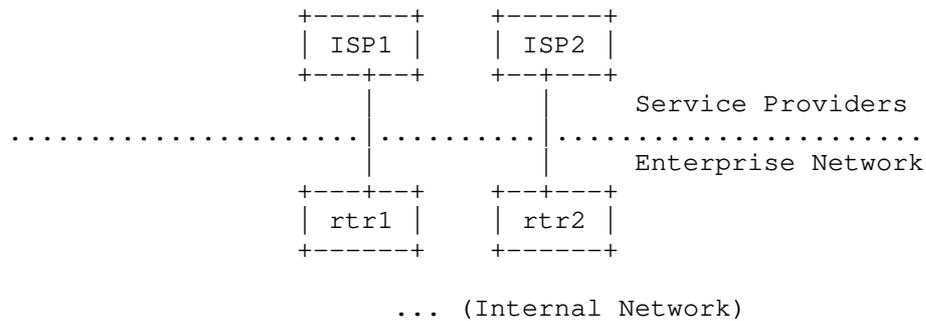   provisioning domain.

```
                +------+   +------+
                | ISP1 |   | ISP2 |
                +---+--+   +--+---+
                    |         |     Service Providers
         ...................|.........|.......................
                    |         |     Enterprise Network
                +---+--+   +--+---+
                | rtr1 |   | rtr2 |
                +------+   +------+

                    ... (Internal Network)
```

          Figure 4: Multi-homed Enterprise Network (Multiple CPEs, Multiple
                          ISPs)

4.4. Multi-homed Enterprise with the Same ISP

   This scenario is a variant of Section 4.2 and Section 4.3 in which
   multi-homing is provided by the same ISP (i.e., same provisioning
   domain).

5. DOTS Deployment Considerations

   Table 1 provides some sample (non-exhaustive) deployment schemes to
   illustrate how DOTS agents may be deployed for each of the scenarios
   introduced in Section 4.

   +-------------------------+------------------------+-------------+
   | Scenario | DOTS client | DOTS |
   | | | gateway |
   +-------------------------+------------------------+-------------+
   | Residential CPE | CPE | N/A |
   +-------------------------+------------------------+-------------+
   | Single CPE, Multiple| internal hosts or CPE | CPE |
   | provisioning domains | | |
   +-------------------------+------------------------+-------------+
   | Multiple CPEs, Multiple| internal hosts or all | CPEs (rtr1 |
   | provisioning domains | CPEs (rtr1 and rtr2) | and rtr2) |
   +-------------------------+------------------------+-------------+
Multi-homed enterprise,	internal hosts or all	CPEs (rtr1
Single provisioning	CPEs (rtr1 and rtr2)	and rtr2)
domain		
   +-------------------------+------------------------+-------------+

                   Table 1: Sample Deployment Cases

   These deployment schemes are further discussed in the following sub-
   sections.

5.1. Residential CPE

   Figure 5 depicts DOTS signaling sessions that are required to be
   established between a DOTS client (C) and DOTS servers (S1, S2) in
   the context of the scenario described in Section 4.1.

   The DOTS client MUST resolve the DOTS server's name provided by a
   provisioning domain ([I-D.boucadair-dots-server-discovery]) using the
   DNS servers learned from the same provisioning domain. The DOTS
   client MUST use the source address selection algorithm defined in
   [RFC6724] to select the candidate source addresses to contact each of
   these DOTS servers. DOTS signaling sessions must be established and
   maintained with each of the DOTS servers because the mitigation scope
   of these servers is restricted. The DOTS client SHOULD use the

certificate provisioned by a provisioning domain to authenticate
itself to the DOTS server provided by the same provisioning domain.
When conveying a mitigation request to protect the attack target(s),
the DOTS client among the DOTS servers available MUST select a DOTS
server whose network has assigned the prefixes from which target
prefixes and target IP addresses are derived. For example,
mitigation request to protect target resources bound to a PA IP
address/prefix cannot be honored by an provisioning domain other than
the one that owns those addresses/prefixes. Consequently, Typically,
if a CPE detects a DDoS attack on all its network attachments, it
must contact both DOTS servers for mitigation. Nevertheless, if the
DDoS attack is received from one single network, then only the DOTS
server of that network must be contacted.

The DOTS client MUST be able to associate a DOTS server with each
provisioning domain. For example, if the DOTS client is provisioned
with S1 using DHCP when attaching to a first network and with S2
using Protocol Configuration Option (PCO) when attaching to a second
network, the DOTS client must record the interface from which a DOTS
server was provisioned. DOTS signaling session to a given DOTS
server must be established using the interface from which the DOTS
server was provisioned.

```
                                        +--+
                         -----------|S1|
                        /             +--+
                       /
                      /
                     /
              +---+/
              | C |
              +---+\
                    \
                     \
                      \             +--+
                       -----------|S2|
                                    +--+
```

         Figure 5: DOTS associations for a multihomed residential CPE

5.2. Multi-homed Enterprise: Single CPE, Multiple Upstream ISPs

   Figure 6 illustrates a first set of DOTS associations that can be
   established with a DOTS gateway is enabled in the context of the
   scenario described in Section 4.2. This deployment is characterized
   as follows:

   o One of more DOTS clients are enabled in hosts located in the
      internal network.

o A DOTS getaway is enabled to aggregate/relay the requests to
   upstream DOTS servers.

When PA addresses/prefixes are in used, the same considerations
discussed in Section 5.1 are to be followed by the DOTS gateway to
contact its DOTS server(s). The DOTS gateways can be reachable from
DOTS client using a unicast or anycast address.

Nevertheless, when PI addresses/prefixes are assigned, the DOTS
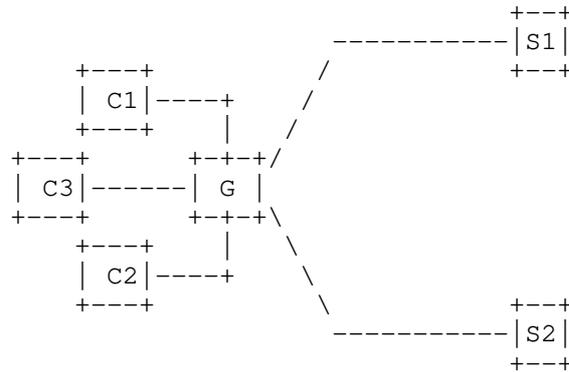gateway MUST sent the same request to all its DOTS servers.

```
                                          +--+
                                 ----------|S1|
            +---+              /           +--+
            | C1|----+    /
            +---+    |   /
      +---+      +-+-+/
      | C3|------| G |
      +---+      +-+-+\
         +---+    |   \
         | C2|----+     \
         +---+           \        +--+
                          ----------|S2|
                                    +--+
```

Figure 6: Multiple DOTS Clients, Single DOTS Gateway, Multiple DOTS
                            Servers

An alternate deployment model is depicted in Figure 7. This
deployment assumes that:

o One of more DOTS clients are enabled in hosts located in the
   internal network. These DOTS client may use
   [I-D.boucadair-dots-server-discovery] to discover its DOTS
   server(s).

o These DOTS clients communicate directly with upstream DOTS
   servers.

If PI addresses/prefixes are in use, the DOTS client can send the
mitigation request for all its PI addresses/prefixes to any one of
the DOTS servers. The use of anycast addresses is NOT RECOMMENDED.

If PA addresses/prefxies are used, the same considerations discussed
in Section 5.1 are to be followed by the DOTS clients. Because DOTS
clients are not located on the CPE and multiple addreses/prefixes may
not be assigned to the DOTS client (IPv4 context, typically), some
complications arise to steer the traffic to the appropriate DOTS

server using the appropriate source IP address. These complications
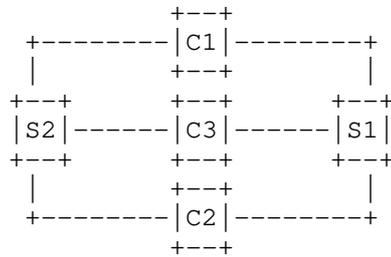discussed in [RFC4116] are not specific to DOTS .

```
                        +--+
            +--------|C1|--------+
            |           +--+           |
     +--+        +--+        +--+
     |S2|------|C3|------|S1|
     +--+        +--+        +--+
            |           +--+           |
            +--------|C2|--------+
                        +--+
```

              Figure 7: Multiple DOTS Clients, Multiple DOTS Servers

Another deployment approach is to enable many DOTS clients; each of
them responsible to handle communication with a specific DOTS server
(see Figure 8). Each DOTS client is provided with policies (e.g.,
prefix filter) that will trigger DOTS communications with the DOTS
servers. The CPE MUST select the appropriate source IP address when
forwarding DOTS messages received from an internal DOTS client. If
anycast addresses are used to reach DOTS servers, the CPE may not be
able to select the appropriate provisioning domain to which the
mitigation request should be forwarded. As a consequence, the
request may not be forwarded to the appropriate DOTS server.

```
                        +--+
            +--------|C1|
            |           +--+
     +--+        +--+        +--+
     |S2|        |C2|------|S1|
     +--+        +--+        +--+
```

                     Figure 8: Single Homed DOTS Clients

5.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

   The deployments depicted in Figure 7 and Figure 8 apply also for the
   scenario described in Section 4.3. One specific problem for this
   scenario is to select the appropriate exit router when contacting a
   given DOTS server.

   An alternative deployment scheme is shown in Figure 9:

   o DOTS clients are enabled in hosts located in the internal network.

   o A DOTS gateway is enabled in each CPE (rtr1, rtr2).

o Each of these DOTS gateways communicate with the DOTS server of
   the provisioning domain.

When PI addresses/prefixes are used, DOTS clients can contact any of
the DOTS gateways to send a DOTS message. DOTS gateway will then
relay the request to the DOTS server. Note that the use of anycast
addresses is NOT RECOMMENDED to establish DOTS signaling sessions
between DOTS client and DOTS gateways.

When PA addresses/prefixes are used, but no filter rules are provided
to DOTS clients, these later MUST contact all DOTS gateways
simultaneously to send a DOTS message. Upon receipt of a request by
a DOTS gateway, it MUST check whether the request is to be forwarded
upstream or be rejected.

When PA addresses/prefixes are used, but specific filter rules are
provided to DOTS clients using some means that are out of scope,
these later MUST select the appropriate DOTS gateway to be contacted.
The use of anycast is NOT RECOMMENDED to reach DOTS gateways.

```
                              +---+
                 +-----------| C1|----+
                 |            +---+    |
  +--+        +-+-+        +---+       +-+-+        +--+
  |S2|------|G2 |------| C3|------|G1 |------|S1|
  +--+        +-+-+        +---+       +-+-+        +--+
                 |            +---+    |
                 +-----------| C2|----+
                              +---+
```

        Figure 9: Multiple DOTS Clients, Multiple DOTS Gateways, Multiple
                              DOTS Servers

5.4. Multi-homed Enterprise: Single ISP

The key difference of the scenario described in Section 4.4 compared
to the other scenarios is that multi-homing is provided by the same
ISP. Concretely, that ISP can decided to provision the enterprise
network with:

1. The same DOTS server for all network attachments.

2. Distinct DOTS servers for each network attachment. These DOTS
    servers needs to coordinate when a mitigation action is received
    from the enterprise network.

In both cases, DOTS agents enabled within the enterprise network may
decide to select one or all network attachments to place DOTS
mitigation requests.

6. Security Considerations

DOTS-related security considerations are discussed in Section 4 of
[I-D.ietf-dots-architecture].

TBD: In Home networks, if EST is used then how will the DOTS gateway
(EST client) be provisioned with credentials for initial enrolment
(see Section 2.2 in RFC 7030).

7. IANA Considerations

This document does not require any action from IANA.

8. Acknowledgements

Thanks to Roland Dobbins and Nik Teague for sharing their comments on
the mailing list.

Thanks to Kirill Kasavchenko for the comments.

9. References

9.1. Normative References

[I-D.ietf-dots-architecture]
          Mortensen, A., Andreasen, F., K, R.,
          christopher_gray3@cable.comcast.com, c., Compton, R., and
          N. Teague, "Distributed-Denial-of-Service Open Threat
          Signaling (DOTS) Architecture", draft-ietf-dots-
          architecture-07 (work in progress), September 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>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
          "Default Address Selection for Internet Protocol Version 6
          (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
          <https://www.rfc-editor.org/info/rfc6724>.

[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

   [I-D.boucadair-dots-server-discovery]
             Boucadair, M., K, R., and P. Patil, "Distributed-Denial-
             of-Service Open Threat Signaling (DOTS) Server Discovery",
             draft-boucadair-dots-server-discovery-04 (work in
             progress), April 2018.

   [I-D.ietf-dots-data-channel]
             Boucadair, M., K, R., Nishizuka, K., Xia, L., Patil, P.,
             Mortensen, A., and N. Teague, "Distributed Denial-of-
             Service Open Threat Signaling (DOTS) Data Channel
             Specification", draft-ietf-dots-data-channel-22 (work in
             progress), September 2018.

   [I-D.ietf-dots-signal-channel]
             K, R., Boucadair, M., Patil, P., Mortensen, A., and N.
             Teague, "Distributed Denial-of-Service Open Threat
             Signaling (DOTS) Signal Channel Specification", draft-
             ietf-dots-signal-channel-25 (work in progress), September
             2018.

   [I-D.ietf-dots-use-cases]
             Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
             Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
             Open Threat Signaling", draft-ietf-dots-use-cases-16 (work
             in progress), July 2018.

   [RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-
             Multihoming Architectures", RFC 3582,
             DOI 10.17487/RFC3582, August 2003,
             <https://www.rfc-editor.org/info/rfc3582>.

   [RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V.
             Gill, "IPv4 Multihoming Practices and Limitations",
             RFC 4116, DOI 10.17487/RFC4116, July 2005,
             <https://www.rfc-editor.org/info/rfc4116>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
             Denial-of-Service Considerations", RFC 4732,
             DOI 10.17487/RFC4732, December 2006,
             <https://www.rfc-editor.org/info/rfc4732>.

   [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain
             Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015,
             <https://www.rfc-editor.org/info/rfc7556>.

   [RFC8043]   Sarikaya, B. and M. Boucadair, "Source-Address-Dependent
               Routing and Source Address Selection for IPv6 Hosts:
               Overview of the Problem Space", RFC 8043,
               DOI 10.17487/RFC8043, January 2017,
               <https://www.rfc-editor.org/info/rfc8043>.

Authors' Addresses

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

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

   Email: TirumaleswarReddy_Konda@McAfee.com

Network Working Group   M. Boucadair
Internet-Draft   Orange
Intended status: Standards Track   T. Reddy
Expires: April 21, 2018   McAfee
                                                              P. Patil
                                                                Cisco
                                                     October 18, 2017

        Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server
                                  Discovery
                  draft-boucadair-dots-server-discovery-03

Abstract

   It may not be possible for a network to determine the cause for an
   attack, but instead just realize that some resources seem to be under
   attack. To fill that gap, Distributed-Denial-of-Service Open Threat
   Signaling (DOTS) allows a network to inform a DOTS server that it is
   under a potential attack so that appropriate mitigation actions are
   undertaken.

   This document specifies mechanisms to configure nodes with DOTS
   servers.

Table of Contents

1. Introduction

   In many deployments, it may not be possible for a network to
   determine the cause for a distributed Denial-of-Service (DoS) attack
   [RFC4732], but instead just realize that some resources seem to be
   under attack. To fill that gap, the IETF is specifying an
   architecture, called DDoS Open Threat Signaling (DOTS)
   [I-D.ietf-dots-architecture], in which a DOTS client can inform a
   DOTS server that the network is under a potential attack and that
   appropriate mitigation actions are required. Indeed, because the
   lack of a common method to coordinate a real-time response among
   involved actors and network domains inhibits the effectiveness of
   DDoS attack mitigation, DOTS protocol is meant to carry requests for
   DDoS attack mitigation, thereby reducing the impact of an attack and
   leading to more efficient defensive actions.
   [I-D.ietf-dots-use-cases] identifies a set of scenarios for DOTS.

   The basic high-level DOTS architecture is illustrated in Figure 1
   ([I-D.ietf-dots-architecture]):

```
      +-----------+                +-------------+
      | Mitigator | ~~~~~~~~~~ | DOTS Server |
      +-----------+                +-------------+
                                          |
                                          |
                                          |
      +---------------+          +-------------+
      | Attack Target | ~~~~~ | DOTS Client |
      +---------------+          +-------------+
```
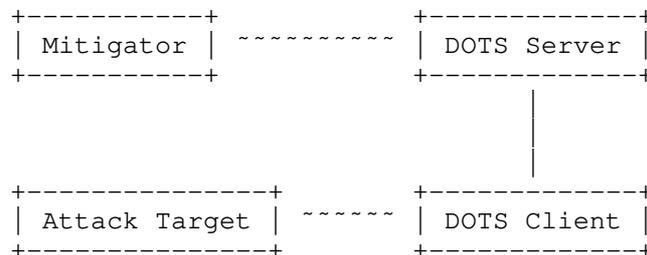
                    Figure 1: Basic DOTS Architecture

   [I-D.ietf-dots-architecture] specifies that the DOTS client may be
   provided with a list of DOTS servers; each associated with one or
   more IP addresses. These addresses may or may not be of the same
   address family. The DOTS client establishes one or more DOTS
   sessions by connecting to the provided DOTS server addresses. The
   logic for connecting to one or multiple IP addresses is out of scope
   of this document.

This document specifies methods for DOTS clients to discover their
DOTS server(s). The rationale for specifying multiple discovery
mechanisms is discussed in Section 4.

Considerations for the selection of DOTS server(s) by multi-homed
DOTS clients is out of scope; the reader should refer to
[I-D.boucadair-dots-multihoming] for more details.

Likewise, happy eyeballs considerations for DOTS are out of scope.
The reader should refer to Section 4 of
[I-D.ietf-dots-signal-channel].

2. Requirements Language

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

3. Terminology

This document makes use of the following terms:

o DDoS: A distributed Denial-of-Service attack, in which traffic
   originating from multiple sources are directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability of servers, services, applications, and/or other
   functionality of an attack target.
o DHCP refers to both DHCPv4 [RFC2131] and DHCPv6 [RFC3315].
o DHCP client denotes a node that initiates requests to obtain
   configuration parameters from one or more DHCP servers.
o DHCP server refers to a node that responds to requests from DHCP
   clients.
o DOTS client: A DOTS-aware software module responsible for
   requesting attack response coordination with other DOTS-aware
   elements.
o DOTS server: A DOTS-aware software module handling and responding
   to messages from DOTS clients. The DOTS server should enable
   mitigation on behalf of the DOTS client, if requested, by
   communicating the DOTS client's request to the mitigator and
   returning selected mitigator feedback to the requesting DOTS
   client. A DOTS server may also be a mitigator.
o DOTS gateway: A DOTS-aware software module that is logically
   equivalent to a DOTS client back-to-back with a DOTS server.

Furthermore, the reader should be familiar with other terms defined
in [I-D.ietf-dots-architecture] and [RFC3958].

4. Why Multiple Discovery Mechanisms?

   It is tempting to specify one single discovery mechanism for DOTS.
   Nevertheless, the analysis of the various use cases sketched in
   [I-D.ietf-dots-use-cases] reveals that it is unlikely that one single
   discovery method can be suitable for all the sample deployments
   (Table 1). Concretely:

   o Some of the use cases may allow DOTS clients to have direct
      communications with upstream DOTS servers; that is no DOTS gateway
      is involved. Leveraging on existing features that do not require
      specific feature on the node embedding the DOTS client may ease
      DOTS deployment. Typically, the use of Straightforward-Naming
      Authority Pointer (S-NAPTR) lookups [RFC3958] allows the DOTS
      server administrators provision the preferred DOTS signal channel
      transport protocol between the DOTS client and the DOTS server and
      allows the DOTS client to discover this preference.

   o Resolving a DOTS server domain name offered by the upstream
      transit provider provisioned to a DOTS client into IP address(es)
      require the use of the appropriate DNS resolvers; otherwise,
      resolving those names will fail. The use of protocols such as
      DHCP does allow to associate provisioned DOTS server domain names
      with a list of DNS servers to be used for name resolution.

   o The upstream network provider is not the DDoS mitigation provider
      for some of these use cases. The use of anycast is not
      appropriate for this use case, in particular. It is safe to
      assume that for such deployments, the DOTS server(s) domain name
      is provided during the service subscription (i.e., manual/local
      configuration).

   o Multiple DOTS clients may be enabled within a network (e.g.,
      enterprise network). Automatic means to discover DOTS servers in
      a deterministic manner are interesting from an operational
      standpoint.

   o Some of the use cases may involve a DOTS gateway that is
      responsible for forking requests received from DOTS clients to
      upstream DOTS servers or for selecting the appropriate DOTS
      server. Particularly, the use of anycast may simplify the
      operations within the enterprise network to discover a DOTS
      gateway, if the enterprise network is single-homed.

   o Many use cases discussed in [I-D.ietf-dots-use-cases] do involve a
      CPE device. Multiple CPEs, connected to distinct network
      providers may even be considered. It is intuitive to leverage on
      existing mechanisms such as discovery using service resolution or

DHCP or anycast to provision the CPE acting as a DOTS client with
the DOTS server(s).

Use Case	Requires a CPE	The Network Provider is also the DDoS Mitigation Provider
End-customer with single or multiple upstream transit provider(s) offering DDoS mitigation services	Yes (Intelligent DDoS mitigation system (IDMS) acting as a DOTS client may be co-located on the CPE)	Yes
End-customer with an overlay DDoS mitigation managed security service provider (MSSP)	Yes (DDOS Detector acting as a DOTS client may be co-located on the CPE)	No
End-customer operating an application or service with an integrated DOTS client	Yes (CPE may act as a DOTS gateway)	Yes/No
End-customer operating a CPE network infrastructure device with an integrated DOTS client	Yes (CPE acts as a DOTS client)	Yes
Suppression of outbound DDoS traffic originating from a consumer broadband access network	Yes (CPE acts as a DOTS server)	Yes
DDoS Orchestration	No	N/A

Table 1: Summary of DOTS Use Cases

Consequently, this document describes the following mechanisms for
discovery:

   o A resolution mechanism based on straightforward Naming Authority
      Pointer (S-NAPTR) resource records in the Domain Name System
      (DNS).

   o DNS Service Discovery.

   o Discovery using DHCP Options.

   o A mechanism based on anycast address for DOTS usage.

5. Discovery Procedure

   A key point in the deployment of DOTS is the ability of network
   operators to be able to configure DOTS clients with the correct
   server information consistently. To accomplish this, operators will
   need a consistent set of ways in which DOTS clients can discover this
   information, and a consistent priority among these options. If some
   devices prefer manual configuration over DNS discovery, while others
   prefer DNS discovery over manual configuration, the result will be a
   process of "whack-a-mole", where the operator must find devices that
   are using the wrong DOTS server, determine how to ensure the devices
   are configured properly, and then reconfigure the device through the
   preferred method.

   All DOTS clients MUST support at least one of the four mechanisms
   below to determine a DOTS server list. All DOTS clients SHOULD
   implement all four, or as many as are practical for any specific
   device, of these ways to discover DOTS servers, in order to
   facilitate the deployment of DOTS in large scale environments:

   1. Explicit configuration:

       * Local/Manual configuration: A DOTS client, will learn the DOTS
          server(s) by means of local or manual DOTS configuration
          (i.e., DOTS servers configured at the system level).
          Configuration discovered from a DOTS client application is
          considered as local configuration. An implementation may give
          the user an opportunity (e.g., by means of configuration file
          options or menu items) to specify DOTS server(s) for each
          address family. These MAY be specified either as IP addresses
          or the DNS name of a DOTS server. When only DOTS server' IP
          addresses are configured, a reference identifier must also be
          configured for authentication purposes.

       * Automatic configuration (e.g., DHCP, an automation system):
          The DOTS client attempts to discover DOTS server(s) names and/
          or addresses from DHCP, as described in Section 9.

2. Service Resolution : The DOTS client attempts to discover DOTS
    server name(s) using service resolution, as specified in
    Section 7.

3. DNS SD: DNS Service Discovery. The DOTS client attempts to
    discover DOTS server name(s) using DNS service discovery, as
    specified in Section 8.

4. Anycast : Send DOTS request to establish a DOTS session with the
    assigned DOTS server anycast address for each combination of
    interface and address family.

Some of these mechanisms imply the use of DNS to resolve the IP
address of the DOTS server, while others imply the IP address of the
relevant DOTS server is obtained directly. Implementation options
may vary on a per device basis, as some devices may not have DNS
capabilities and/or proper configuration.

Clients will prefer information received from the discovery methods
in the order listed.

On hosts with more than one interface or address family (IPv4/v6),
the DOTS server discovery procedure has to be performed for each
combination of interface and address family. A client MAY choose to
perform the discovery procedure only for a desired interface/address
combination if the client does not wish to discover a DOTS server for
all combinations of interface and address family.

The above procedure MUST also be followed by a DOTS gateway.

6. Resolution

Once the DOTS client has retrieved client's DNS domain or discovered
the DOTS server name that needs to be resolved, an S-NAPTR lookup
with 'DOTS' application service and the desired protocol tag is made
to obtain information necessary to connect to the authoritative DOTS
server within the given domain.

This specification defines "DOTS" as an application service tag
(Section 12.3.1) and "signal.udp" (Section 12.3.2), "signal.tcp"
(Section 12.3.3), and "data.tcp" (Section 12.3.4) as application
protocol tags.

In the example below, for domain 'example.net', the resolution
algorithm will result in IP address(es), port, tag and protocol tuples as
follows:

```
example.net.
IN NAPTR 100 10 "" DOTS:signal.udp "" signal.example.net.
IN NAPTR 200 10 "" DOTS:signal.tcp "" signal.example.net.
IN NAPTR 300 10 "" DOTS:data.tcp "" data.example.net.

signal.example.net.
IN NAPTR 100 10 S DOTS:signal.udp "" _dots._signal._udp.example.net.
IN NAPTR 200 10 S DOTS:signal.tcp "" _dots._signal._tcp.example.net.

data.example.net.
IN NAPTR 100 10 S DOTS:data.tcp "" _dots._data._tcp.example.net.

_dots._signal._udp.example.net.
IN SRV   0 0 5000 a.example.net.

_dots._signal._tcp.example.net.
IN SRV   0 0 5001 a.example.net.

_dots._data._tcp.example.net.
IN SRV   0 0 5002 a.example.net.

a.example.net.
IN AAAA  2001:db8::1
```

Order	Protocol	IP address	Port	Tag
1	UDP	2001:db8::1	5000	Signal
2	TCP	2001:db8::1	5001	Signal
3	TCP	2001:db8::1	5002	Data

If no DOTS-specific S-NAPTR records can be retrieved, the discovery
procedure fails for this domain name (and the corresponding interface
and IP protocol version). If more domain names are known, the
discovery procedure MAY perform the corresponding S-NAPTR lookups
immediately. However, before retrying a lookup that has failed, a
DOTS client MUST wait a time period that is appropriate for the
encountered error (e.g., NXDOMAIN, timeout, etc.).

7. Discovery using Service Resolution

    This mechanism is performed in two steps:

    1. A DNS domain name is retrieved for each combination of interface
        and address family.

    2. Retrieved DNS domain names are then used for S-NAPTR lookups.
        Further DNS lookups may be necessary to determine DOTS server IP
        address(es).

7.1. Retrieving Domain Name

    A DOTS client has to determine the domain in which it is located.
    The following section describes the means to obtain the domain name
    from DHCP. Other means of retrieving domain names may be used, which
    are outside the scope of this document, e.g., local configuration.

    Implementations MAY allow the user to specify a default name that is
    used, if no specific name has been configured.

7.1.1. DHCP

    DHCP can be used to determine the domain name related to an
    interface's point of network attachment. Network operators may
    provide the domain name to be used for service discovery within an
    access network using DHCP. Sections 3.2 and 3.3 of [RFC5986] define
    DHCP IPv4 and IPv6 access network domain name options,
    OPTION_V4_ACCESS_DOMAIN and OPTION_V6_ACCESS_DOMAIN respectively, to
    identify a domain name that is suitable for service discovery within
    the access network.

    For IPv4, the discovery procedure MUST request the access network
    domain name option in a Parameter Request List option, as described
    in [RFC2131]. [RFC2132] defines the DHCP IPv4 domain name option;
    while this option is less suitable, a client MAY request for it if
    the access network domain name defined in [RFC5986] is not available.

    For IPv6, the discovery procedure MUST request for the access network
    domain name option in an Options Request Option (ORO) within an
    Information-request message, as described in [RFC3315].

    If neither option can be retrieved the procedure fails for this
    interface. If a result can be retrieved it will be used as an input
    for S-NAPTR resolution discussed in Section 6.

8. DNS Service Discovery

   DNS-based Service Discovery (DNS-SD) [RFC6763] and Multicast DNS
   (mDNS) [RFC6762] provide generic solutions for discovering services.
   DNS-SD/mDNS define a set of naming rules for certain DNS record types
   that they use for advertising and discovering services.

8.1. DNS-SD

   Section 4.1 of [RFC6763] specifies that a service instance name in
   DNS-SD has the following structure:

   <Instance> . <Service> . <Domain>

   The <Domain> portion specifies the DNS sub-domain where the service
   instance is registered. It may be "local.", indicating the mDNS
   local domain, or it may be a conventional domain name such as
   "example.com.".

   The <Service> portion of the DOTS service instance name MUST be
   "_dots._signal._udp" or "_dots._signal._tcp" or "_dots._data._tcp".

8.2. mDNS

   A DOTS client can proactively discover DOTS servers being advertised
   in the site by multicasting a PTR query to one or all of the
   following:

   o "_dots._signal._udp.local."

   o "_dots._signal._tcp.local."

   o "_dots._data._tcp.local."

   A DOTS server can send out gratuitous multicast DNS answer packets
   whenever it starts up, wakes from sleep, or detects a change in
   network configuration. DOTS clients receive these gratuitous packets
   and cache information contained in it.

9. DHCP Options for DOTS

   As reported in Section 1.7.2 of [RFC6125]:

      "few certification authorities issue server certificates based on
      IP addresses, but preliminary evidence indicates that such
      certificates are a very small percentage (less than 1%) of issued
      certificates".

In order to allow for PKIX-based authentication between a DOTS client
and server while accommodating for the current best practices for
issuing certificates, this document allows for configuring names to
DOTS clients. These names can be used for two purposes: to retrieve
the list of IP addresses of a DOTS server or to be presented as a
reference identifier for authentication purposes.

Defining the option to include a list of IP addresses would avoid a
dependency on an underlying name resolution, but that design requires
to also supply a name for PKIX-based authentication purposes.

9.1. DHCPv6 DOTS Options

9.1.1. Format of DOTS Reference Identifier Option

The DHCPv6 DOTS option is used to configure a name of the DOTS
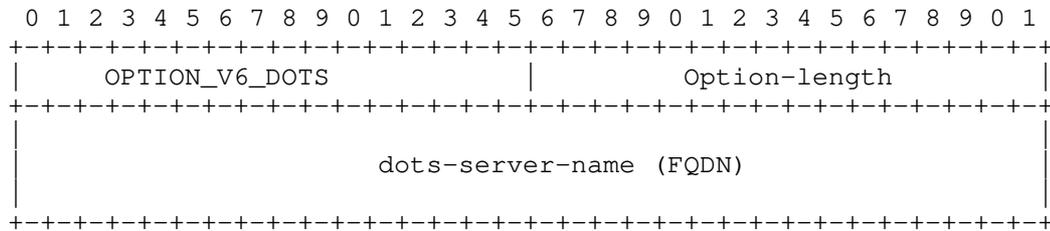server. The format of this option is shown in Figure 2.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        OPTION_V6_DOTS          |         Option-length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                     dots-server-name (FQDN)                   |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

              Figure 2: DHCPv6 DOTS Reference Identifier option

The fields of the option shown in Figure 2 are as follows:

o Option-code: OPTION_V6_DOTS_RI (TBA1, see Section 12.1)
o Option-length: Length of the dots-server-name field in octets.
o dots-server-name: A fully qualified domain name of the DOTS
   server. This field is formatted as specified in Section 8 of
   [RFC3315].

An example of the dots-server-name encoding is shown in Figure 3.
This example conveys the FQDN "dots.example.com.".

```
    +------+------+------+------+------+------+------+------+------+
    | 0x04 |  d   |  o   |  t   |  s   | 0x07 |  e   |  x   |  a   |
    +------+------+------+------+------+------+------+------+------+
    |  m   |  p   |  l   |  e   | 0x03 |  c   |  o   |  m   | 0x00 |
    +------+------+------+------+------+------+------+------+------+
```

         Figure 3: An example of the dots-server-name encoding

9.1.2. Format Format of DOTS Address Option

   The DHCPv6 DOTS option can be used to configure a list of IPv6
   addresses of a DOTS server. The format of this option is shown in
   Figure 4.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         OPTION_V6_DOTS         |           Option-length       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                                                               |
    |                      DOTS ipv6-address                        |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                                                               |
    |                      DOTS ipv6-address                        |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             ...                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 4: DHCPv6 DOTS Address option

   The fields of the option shown in Figure 4 are as follows:

   o Option-code: OPTION_V6_DOTS_ADDRESS (TBA2, see Section 12.1)
   o Option-length: Length of the 'DOTS ipv6-address(es)' field in
      octets. MUST be a multiple of 16.
   o DOTS ipv6-address: Includes one or more IPv6 addresses [RFC4291]
      of the DOTS server to be used by the DOTS client.

      Note, IPv4-mapped IPv6 addresses (Section 2.5.5.2 of [RFC4291])
      are allowed to be included in this option.

   To return more than one DOTS servers to the requesting DHCPv6 client,
   the DHCPv6 server returns multiple instances of OPTION_V6_DOTS.

9.1.3. DHCPv6 Client Behavior

   DHCP clients MAY request options OPTION_V6_DOTS_RI and
   OPTION_V6_DOTS_ADDRESS, as defined in [RFC3315], Sections 17.1.1,
   18.1.1, 18.1.3, 18.1.4, 18.1.5, and 22.7. As a convenience to the
   reader, it is mentioned here that the DHCP client includes the
   requested option codes in the Option Request Option.

If the DHCP client receives more than one instance of
OPTION_V6_DOTS_RI (resp. OPTION_V6_DOTS_ADDRESS) option, it MUST use
only the first instance of that option.

If the DHCP client receives both OPTION_V6_DOTS_RI and
OPTION_V6_DOTS_ADDRESS, the content of OPTION_V6_DOTS_RI is used as
reference identifier for authentication purposes (e.g., PKIX
[RFC6125]), while the addresses included in OPTION_V6_DOTS_ADDRESS
are used to reach the DOTS server. In other words, the name conveyed
in OPTION_V6_DOTS_RI MUST NOT be passed to underlying resolution
library in the presence of OPTION_V6_DOTS_ADDRESS in a response.

If the DHCP client receives OPTION_V6_DOTS_RI only, but
OPTION_V6_DOTS_RI option contains more than one name, as
distinguished by the presence of multiple root labels, the DHCP
client MUST use only the first name. Once the name is validated
(Section 8 of [RFC3315]), the name is passed to a name resolution
library. Moreover, that name is also used as a reference identifier
for authentication purposes.

If the DHCP client receives OPTION_V6_DOTS_ADDRESS only, the
address(es) included in OPTION_V6_DOTS_ADDRESS is used to reach the
DOTS server. In addition, these addresses can be used as identifiers
for authentication.

9.2. DHCPv4 DOTS Options

9.2.1. Format of DOTS Reference Identifier Option

The DHCPv4 DOTS option is used to configure a name of the DOTS
server. The format of this option is illustrated in Figure 5.

```
        Code  Length   DOTS server name
       +-----+-----+-----+-----+-----+-----+-----+--
       | TBA |  n  | s1  | s2  | s3  | s4  | s5  | ...
       +-----+-----+-----+-----+-----+-----+-----+--
```

The values s1, s2, s3, etc. represent the domain name labels in the
domain name encoding.

             Figure 5: DHCPv4 DOTS Reference Identifier option

The fields of the option shown in Figure 5 are as follows:

o Code: OPTION_V4_DOTS_RI (TBA3, see Section 12.2);
o Length: Includes the length of the "DOTS server name" field in
   octets; the maximum length is 255 octets.

   o DOTS server name: The domain name of the DOTS server. This field
      is formatted as specified in Section 8 of [RFC3315].

9.2.2. Format Format of DOTS Address Option

   The DHCPv4 DOTS option can be used to configure a list of IPv4
   addresses of a DOTS server. The format of this option is illustrated
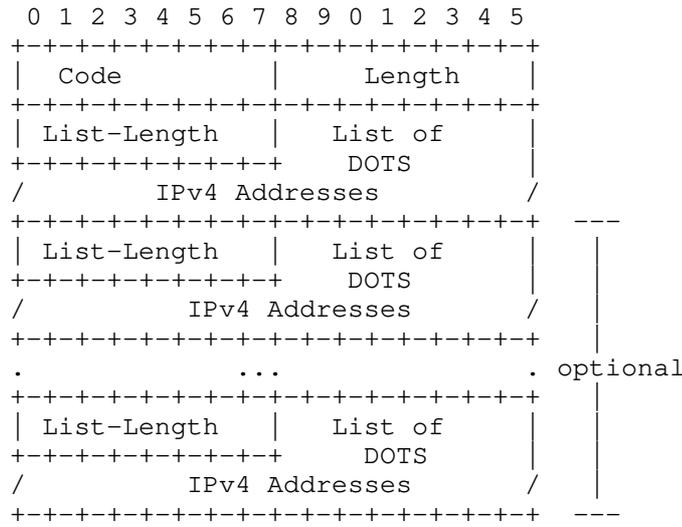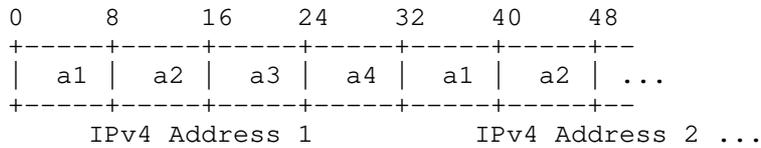   in Figure 6.

```
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Code      |     Length    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | List-Length   |    List of    |
   +-+-+-+-+-+-+-+-+     DOTS       |
   /          IPv4 Addresses       /
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---
   | List-Length   |    List of    |   |
   +-+-+-+-+-+-+-+-+     DOTS       |   |
   /          IPv4 Addresses       /   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   .               ...                 . optional
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   | List-Length   |    List of    |   |
   +-+-+-+-+-+-+-+-+     DOTS       |   |
   /          IPv4 Addresses       /   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---
```

                Figure 6: DHCPv4 DOTS Address option

   The fields of the option shown in Figure 6 are as follows:

   o Code: OPTION_V4_DOTS_ADDRESS (TBA4, see Section 12.2);
   o Length: Length of all included data in octets. The minimum length
      is 5.
   o List-Length: Length of the "List of DOTS IPv4 Addresses" field in
      octets; MUST be a multiple of 4.
   o List of DOTS IPv4 Addresses: Contains one or more IPv4 addresses
      of the DOTS server to be used by the DOTS client. The format of
      this field is shown in Figure 7.
   o OPTION_V4_DOTS can include multiple lists of DOTS IPv4 addresses;
      each list is treated separately as it corresponds to a given DOTS
      server.

      When several lists of DOTS IPv4 addresses are to be included,
      "List-Length" and "DOTS IPv4 Addresses" fields are repeated.

```
      0     8    16    24    32    40    48
      +-----+-----+-----+-----+-----+-----+--
      | a1  | a2  | a3  | a4  | a1  | a2  | ...
      +-----+-----+-----+-----+-----+-----+--
          IPv4 Address 1          IPv4 Address 2 ...
```

   This format assumes that an IPv4 address is encoded as a1.a2.a3.a4.

              Figure 7: Format of the List of DOTS IPv4 Addresses

   OPTION_V4_DOTS is a concatenation-requiring option. As such, the
   mechanism specified in [RFC3396] MUST be used if OPTION_V4_DOTS
   exceeds the maximum DHCPv4 option size of 255 octets.

9.2.3. DHCPv4 Client Behavior

   To discover a DOTS server, the DHCPv4 client MUST include both
   OPTION_V4_DOTS_RI and OPTION_V4_DOTS_ADDRESS in a Parameter Request
   List Option [RFC2132].

   If the DHCP client receives more than one instance of
   OPTION_V4_DOTS_RI (resp. OPTION_V4_DOTS_ADDRESS) option, it MUST use
   only the first instance of that option.

   If the DHCP client receives both OPTION_V4_DOTS_RI and
   OPTION_V4_DOTS_ADDRESS, the content of OPTION_V6_DOTS_RI is used as
   reference identifier for authentication purposes, while the addresses
   included in OPTION_V4_DOTS_ADDRESS are used to reach the DOTS server.
   In other words, the name conveyed in OPTION_V4_DOTS_RI MUST NOT be
   passed to underlying resolution library in the presence of
   OPTION_V4_DOTS_ADDRESS in a response.

   If the DHCP client receives OPTION_V4_DOTS_RI only, but
   OPTION_V4_DOTS_RI option contains more than one name, as
   distinguished by the presence of multiple root labels, the DHCP
   client MUST use only the first name. Once the name is validated
   (Section 8 of [RFC3315]), the name is passed to a name resolution
   library. Moreover, that name is also used as a reference identifier
   for authentication purposes.

   If the DHCP client receives OPTION_V4_DOTS_ADDRESS only, the
   address(es) included in OPTION_V4_DOTS_ADDRESS is used to reach the
   DOTS server. In addition, these addresses can be used as identifiers
   for authentication.

10. Anycast

   IP anycast can also be used for DOTS service discovery. A packet
   sent to an anycast address is delivered to the 'topologically
   nearest' network interface with the anycast address.

   When a DOTS client requires DOTS services, it attempts to establish a
   signaling session with the assigned anycast address(es) defined in
   Sections 12.4 and 12.5. A DOTS server, that receives a DOTS request
   with an anycast address, SHOULD redirect the DOTS client to the
   appropriate DOTS unicast server(s) using the mechanism described in
   Section 5.5 of [I-D.ietf-dots-signal-channel], unless it is
   configured otherwise. Indeed, a DOTS server SHOULD be configurable
   to maintain all DOTS communications using anycast. DOTS redirect is
   not made mandatory because the use of anycast is not problematic for
   some deployment scenarios such as an enterprise network deploying one
   single DOTS gateway connected to one single network provider.

   [I-D.boucadair-dots-multihoming] identifies a set of deployment
   schemes in which the use of anycast is not recommended.

11. Security Considerations

   DOTS-related security considerations are discussed in Section 4 of
   [I-D.ietf-dots-architecture] is to be considered. DOTS agents must
   authenticate each other using (D)TLS before a DOTS session is
   considered valid.

   If the DOTS client is explicitly configured with DOTS server(s) then
   the DOTS client can also be explicitly configured with credentials to
   authenticate the DOTS server.

   The CPE device acting as a DOTS client MAY use Bootstrapping Remote
   Secure Key Infrastructures (BRSKI) discussed in
   [I-D.ietf-anima-bootstrapping-keyinfra] to automatically bootstrap
   using the vendor installed X.509 certificate, in combination with a
   domain registrar provided by the upstream transit provider and
   vendor's authorizing service. The CPE device authenticates to the
   upstream transit provider using the vendor installed X.509
   certificate and the upstream transit provider validates the vendor
   installed certificate on the CPE device using the Manufacturer
   Authorized Signing Authority (MASA) service. If authentication is
   successful then the CPE device can request and get a voucher from the
   MASA service via the domain registrar. The voucher is signed by the
   MASA service and includes the upstream transit provider's trust
   anchor certificate. The CPE device validates the signed voucher
   using the manufacturer installed trust anchor associated with the
   vendor's selected MASA service and stores the upstream transit

provider's trust anchor certificate. The CPE device then uses
Enrollment over Secure Transport (EST) [RFC7030] for certificate
enrollment (Section 3.8 in [I-D.ietf-anima-bootstrapping-keyinfra]).
The DOTS client on the CPE device can authenticate to the DOTS server
using the certificate provisioned by the EST server and the DOTS
client can validate the DOTS server certificate using the upstream
transit provider's trust anchor certificate it had received in the
voucher.

11.1. DHCP

The security considerations in [RFC2131] and [RFC3315] are to be
considered.

11.2. Service Resolution

The primary attack against the methods described in Section 7 is one
that would lead to impersonation of a DOTS server. An attacker could
attempt to compromise the S-NAPTR resolution. The use of mutual
authentication makes it difficult to redirect a DOTS client to an
illegitimate DOTS server.

11.3. DNS Service Discovery

Since DNS-SD is just a specification for how to name and use records
in the existing DNS system, it has no specific additional security
requirements over and above those that already apply to DNS queries
and DNS updates. For DNS queries, DNS Security Extensions (DNSSEC)
[RFC4033] SHOULD be used where the authenticity of information is
important. For DNS updates, secure updates [RFC2136][RFC3007] SHOULD
generally be used to control which clients have permission to update
DNS records.

For mDNS, in addition to what has been described above, a principal
security threat is a security threat inherent to IP multicast routing
and any application that runs on it. A rogue system can advertise
that it is a DOTS server. Discovery of such rogue systems as DOTS
servers, in itself, is not a security threat if the DOTS client
authenticates the discovered DOTS servers.

11.4. Anycast

Anycast-related security considerations are discussed in [RFC4786]
and [RFC7094].

12. IANA Considerations

   IANA is requested to allocate the SRV service name of "_dots._signal"
   for DOTS signal channel over UDP or TCP, and the service name of
   "_dots._data" for DOTS data channel over TCP.

12.1. DHCPv6 Option

   IANA is requested to assign the following new DHCPv6 Option Code in
   the registry maintained in http://www.iana.org/assignments/
   dhcpv6-parameters:

                          Option Name Value
                    --------------------- -----
                        OPTION_V6_DOTS_RI TBA1
                   OPTION_V6_DOTS_ADDRESS TBA2

12.2. DHCPv4 Option

   IANA is requested to assign the following new DHCPv4 Option Code in
   the registry maintained in http://www.iana.org/assignments/bootp-
   dhcp-parameters/:

Option Name	Value	Data length	Meaning
OPTION_V4_DOTS_RI	TBA3	Variable; the maximum length is 255 octets.	Includes the name of the DOTS server.
OPTION_V4_DOTS_ADDRESS	TBA4	Variable; the minimum length is 5.	Includes one or multiple lists of DOTS IP addresses; each list is treated as a separate DOTS server.

12.3. Application Service & Application Protocol Tags

   This document requests IANA to make the following allocations from
   the registry available at: https://www.iana.org/assignments/s-naptr-
   parameters/s-naptr-parameters.xhtml.

12.3.1. DOTS Application Service Tag Registration

   o Application Protocol Tag: DOTS

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.2. signal.udp Application Protocol Tag Registration

   o Application Protocol Tag: signal.udp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.3. signal.tcp Application Protocol Tag Registration

   o Application Protocol Tag: signal.tcp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.4. data.tcp Application Protocol Tag Registration

   o Application Protocol Tag: data.tcp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.4. IPv4 Anycast

   IANA has assigned a single IPv4 address from the 192.0.0.0/24 prefix
   and registered it in the "IANA IPv4 Special-Purpose Address Registry"
   [RFC6890].

```
+--------------------+-----------------------------------------+
| Attribute          | Value                                   |
+--------------------+-----------------------------------------+
| Address Block      | TBA                                     |
| Name               | Distributed-Denial-of-Service Open Threat|
|                    | Signaling (DOTS) Anycast                |
| RFC                | <this document>                         |
| Allocation Date    | <date of approval of this document>     |
| Termination Date   | N/A                                     |
| Source             | True                                    |
| Destination        | True                                    |
| Forwardable        | True                                    |
| Global             | True                                    |
| Reserved-by-Protocol | False                                 |
+--------------------+-----------------------------------------+
```

12.5. IPv6 Anycast

   IANA has assigned a single IPv6 address from the 2001:0000::/23
   prefix and registered it in the "IANA IPv6 Special-Purpose Address
   Registry" [RFC6890].

```
+--------------------+-----------------------------------------+
| Attribute          | Value                                   |
+--------------------+-----------------------------------------+
| Address Block      | TBA                                     |
| Name               | Distributed-Denial-of-Service Open Threat|
|                    | Signaling (DOTS) Anycast                |
| RFC                | <this document>                         |
| Allocation Date    | <date of approval of this document>     |
| Termination Date   | N/A                                     |
| Source             | True                                    |
| Destination        | True                                    |
| Forwardable        | True                                    |
| Global             | True                                    |
| Reserved-by-Protocol | False                                 |
+--------------------+-----------------------------------------+
```

13. Acknowledgements

   Thanks to Brian Carpenter for the review of the BRSKI text.

   Many thanks to Russ White for the review, comments, and text
   contribution.

14. References

14.1. Normative References

   [I-D.ietf-dots-architecture]
              Mortensen, A., Andreasen, F., Reddy, T.,
              christopher_gray3@cable.comcast.com, c., Compton, R., and
              N. Teague, "Distributed-Denial-of-Service Open Threat
              Signaling (DOTS) Architecture", draft-ietf-dots-
              architecture-04 (work in progress), July 2017.

   [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>.

   [RFC2131] Droms, R., "Dynamic Host Configuration Protocol",
              RFC 2131, DOI 10.17487/RFC2131, March 1997,
              <https://www.rfc-editor.org/info/rfc2131>.

   [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor
              Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,
              <https://www.rfc-editor.org/info/rfc2132>.

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
              C., and M. Carney, "Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
              2003, <https://www.rfc-editor.org/info/rfc3315>.

   [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the
              Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396,
              DOI 10.17487/RFC3396, November 2002,
              <https://www.rfc-editor.org/info/rfc3396>.

   [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application
              Service Location Using SRV RRs and the Dynamic Delegation
              Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958,
              January 2005, <https://www.rfc-editor.org/info/rfc3958>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local
              Location Information Server (LIS)", RFC 5986,
              DOI 10.17487/RFC5986, September 2010,
              <https://www.rfc-editor.org/info/rfc5986>.

   [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,
              DOI 10.17487/RFC6762, February 2013,
              <https://www.rfc-editor.org/info/rfc6762>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [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>.

14.2. Informative References

   [I-D.boucadair-dots-multihoming]
              Boucadair, M. and T. Reddy, "Multi-homing Deployment
              Considerations for Distributed-Denial-of-Service Open
              Threat Signaling (DOTS)", draft-boucadair-dots-
              multihoming-02 (work in progress), October 2017.

   [I-D.ietf-anima-bootstrapping-keyinfra]
              Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
              S., and K. Watsen, "Bootstrapping Remote Secure Key
              Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
              keyinfra-08 (work in progress), October 2017.

   [I-D.ietf-dots-signal-channel]
              Reddy, T., Boucadair, M., Patil, P., Mortensen, A., and N.
              Teague, "Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-
              channel-05 (work in progress), October 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-07 (work
              in progress), July 2017.

   [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>.

   [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>.

   [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>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [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>.

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
              2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
              "Enrollment over Secure Transport", RFC 7030,
              DOI 10.17487/RFC7030, October 2013,
              <https://www.rfc-editor.org/info/rfc7030>.

   [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
              "Architectural Considerations of IP Anycast", RFC 7094,
              DOI 10.17487/RFC7094, January 2014,
              <https://www.rfc-editor.org/info/rfc7094>.

Authors' Addresses

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

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

   Email: TirumaleswarReddy_Konda@McAfee.com

   Prashanth Patil
   Cisco Systems, Inc.

   Email: praspati@cisco.com

       Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server
                                Discovery
                 draft-boucadair-dots-server-discovery-05

Abstract

   It may not be possible for a network to determine the cause for an
   attack, but instead just realize that some resources seem to be under
   attack. To fill that gap, Distributed-Denial-of-Service Open Threat
   Signaling (DOTS) allows a network to inform a DOTS server that it is
   under a potential attack so that appropriate mitigation actions are
   undertaken.

   This document specifies mechanisms to configure nodes with DOTS
   servers.

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

Table of Contents

1. Introduction

   In many deployments, it may not be possible for a network to
   determine the cause for a distributed Denial-of-Service (DoS) attack
   [RFC4732], but instead just realize that some resources seem to be
   under attack. To fill that gap, the IETF is specifying an
   architecture, called DDoS Open Threat Signaling (DOTS)
   [I-D.ietf-dots-architecture], in which a DOTS client can inform a
   DOTS server that the network is under a potential attack and that
   appropriate mitigation actions are required. Indeed, because the
   lack of a common method to coordinate a real-time response among
   involved actors and network domains inhibits the effectiveness of
   DDoS attack mitigation, DOTS protocol is meant to carry requests for
   DDoS attack mitigation, thereby reducing the impact of an attack and
   leading to more efficient defensive actions.
   [I-D.ietf-dots-use-cases] identifies a set of scenarios for DOTS.

   The basic high-level DOTS architecture is illustrated in Figure 1
   ([I-D.ietf-dots-architecture]):

```
         +-----------+                  +-------------+
         | Mitigator | ~~~~~~~~~~ | DOTS Server |
         +-----------+                  +-------------+
                                               |
                                               |
                                               |
         +---------------+          +-------------+
         | Attack Target | ~~~~~ | DOTS Client |
         +---------------+          +-------------+
```
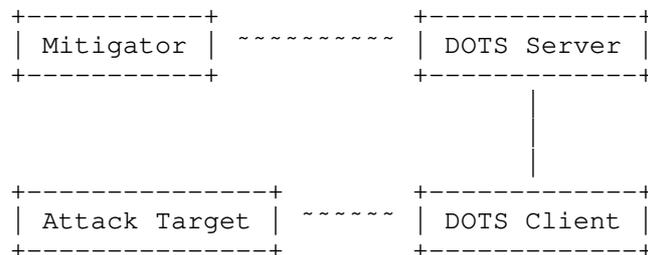
                   Figure 1: Basic DOTS Architecture

   [I-D.ietf-dots-architecture] specifies that the DOTS client may be
   provided with a list of DOTS servers; each associated with one or
   more IP addresses. These addresses may or may not be of the same
   address family. The DOTS client establishes one or more DOTS
   sessions by connecting to the provided DOTS server addresses. The
   logic for connecting to one or multiple IP addresses is out of scope
   of this document.

This document specifies methods for DOTS clients to discover their
DOTS server(s). The rationale for specifying multiple discovery
mechanisms is discussed in Section 4.

Considerations for the selection of DOTS server(s) by multi-homed
DOTS clients is out of scope; the reader should refer to
[I-D.boucadair-dots-multihoming] for more details.

Likewise, happy eyeballs considerations for DOTS are out of scope.
The reader should refer to Section 4 of
[I-D.ietf-dots-signal-channel].

2. Requirements Language

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. Terminology

This document makes use of the following terms:

o DDoS: A distributed Denial-of-Service attack, in which traffic
   originating from multiple sources are directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability of servers, services, applications, and/or other
   functionality of an attack target.
o DHCP refers to both DHCPv4 [RFC2131] and DHCPv6 [RFC3315].
o DHCP client denotes a node that initiates requests to obtain
   configuration parameters from one or more DHCP servers.
o DHCP server refers to a node that responds to requests from DHCP
   clients.
o DOTS client: A DOTS-aware software module responsible for
   requesting attack response coordination with other DOTS-aware
   elements.
o DOTS server: A DOTS-aware software module handling and responding
   to messages from DOTS clients. The DOTS server should enable
   mitigation on behalf of the DOTS client, if requested, by
   communicating the DOTS client's request to the mitigator and
   returning selected mitigator feedback to the requesting DOTS
   client. A DOTS server may also be a mitigator.
o DOTS gateway: A DOTS-aware software module that is logically
   equivalent to a DOTS client back-to-back with a DOTS server.

Furthermore, the reader should be familiar with other terms defined
in [I-D.ietf-dots-architecture] and [RFC3958].

4. Why Multiple Discovery Mechanisms?

   It is tempting to specify one single discovery mechanism for DOTS.
   Nevertheless, the analysis of the various use cases sketched in
   [I-D.ietf-dots-use-cases] reveals that it is unlikely that one single
   discovery method can be suitable for all the sample deployments
   (Table 1). Concretely:

   o Some of the use cases may allow DOTS clients to have direct
      communications with upstream DOTS servers; that is no DOTS gateway
      is involved. Leveraging on existing features that do not require
      specific feature on the node embedding the DOTS client may ease
      DOTS deployment. Typically, the use of Straightforward-Naming
      Authority Pointer (S-NAPTR) lookups [RFC3958] allows the DOTS
      server administrators provision the preferred DOTS signal channel
      transport protocol between the DOTS client and the DOTS server and
      allows the DOTS client to discover this preference.

   o Resolving a DOTS server domain name offered by the upstream
      transit provider provisioned to a DOTS client into IP address(es)
      require the use of the appropriate DNS resolvers; otherwise,
      resolving those names will fail. The use of protocols such as
      DHCP does allow to associate provisioned DOTS server domain names
      with a list of DNS servers to be used for name resolution.

   o The upstream network provider is not the DDoS mitigation provider
      for some of these use cases. The use of anycast is not
      appropriate for this use case, in particular. It is safe to
      assume that for such deployments, the DOTS server(s) domain name
      is provided during the service subscription (i.e., manual/local
      configuration).

   o Multiple DOTS clients may be enabled within a network (e.g.,
      enterprise network). Automatic means to discover DOTS servers in
      a deterministic manner are interesting from an operational
      standpoint.

   o Some of the use cases may involve a DOTS gateway that is
      responsible for forking requests received from DOTS clients to
      upstream DOTS servers or for selecting the appropriate DOTS
      server. Particularly, the use of anycast may simplify the
      operations within the enterprise network to discover a DOTS
      gateway, if the enterprise network is single-homed.

   o Many use cases discussed in [I-D.ietf-dots-use-cases] do involve a
      CPE device. Multiple CPEs, connected to distinct network
      providers may even be considered. It is intuitive to leverage on
      existing mechanisms such as discovery using service resolution or

DHCP or anycast to provision the CPE acting as a DOTS client with
the DOTS server(s).

Use Case	Requires a CPE	The Network Provider is also the DDoS Mitigation Provider
End-customer with single or multiple upstream transit provider(s) offering DDoS mitigation services	Yes (Intelligent DDoS mitigation system (IDMS) acting as a DOTS client may be co-located on the CPE)	Yes
End-customer with an overlay DDoS mitigation managed security service provider (MSSP)	Yes (DDOS Detector acting as a DOTS client may be co-located on the CPE)	No
End-customer operating an application or service with an integrated DOTS client	Yes (CPE may act as a DOTS gateway)	Yes/No
End-customer operating a CPE network infrastructure device with an integrated DOTS client	Yes (CPE acts as a DOTS client)	Yes
Suppression of outbound DDoS traffic originating from a consumer broadband access network	Yes (CPE acts as a DOTS server)	Yes
DDoS Orchestration	No	N/A

Table 1: Summary of DOTS Use Cases

Consequently, this document describes the following mechanisms for
discovery:

o A resolution mechanism based on straightforward Naming Authority
   Pointer (S-NAPTR) resource records in the Domain Name System
   (DNS).

o DNS Service Discovery.

o Discovery using DHCP Options.

o A mechanism based on anycast address for DOTS usage.

5. Discovery Procedure

   A key point in the deployment of DOTS is the ability of network
   operators to be able to configure DOTS clients with the correct
   server information consistently. To accomplish this, operators will
   need a consistent set of ways in which DOTS clients can discover this
   information, and a consistent priority among these options. If some
   devices prefer manual configuration over DNS discovery, while others
   prefer DNS discovery over manual configuration, the result will be a
   process of "whack-a-mole", where the operator must find devices that
   are using the wrong DOTS server, determine how to ensure the devices
   are configured properly, and then reconfigure the device through the
   preferred method.

   All DOTS clients MUST support at least one of the four mechanisms
   below to determine a DOTS server list. All DOTS clients SHOULD
   implement all four, or as many as are practical for any specific
   device, of these ways to discover DOTS servers, in order to
   facilitate the deployment of DOTS in large scale environments:

   1. Explicit configuration:

       * Local/Manual configuration: A DOTS client, will learn the DOTS
          server(s) by means of local or manual DOTS configuration
          (i.e., DOTS servers configured at the system level).
          Configuration discovered from a DOTS client application is
          considered as local configuration. An implementation may give
          the user an opportunity (e.g., by means of configuration file
          options or menu items) to specify DOTS server(s) for each
          address family. These MAY be specified either as IP addresses
          or the DNS name of a DOTS server. When only DOTS server' IP
          addresses are configured, a reference identifier must also be
          configured for authentication purposes.

       * Automatic configuration (e.g., DHCP, an automation system):
          The DOTS client attempts to discover DOTS server(s) names and/
          or addresses from DHCP, as described in Section 9.

   2. Service Resolution : The DOTS client attempts to discover DOTS
       server name(s) using service resolution, as specified in
       Section 7.

   3. DNS SD: DNS Service Discovery. The DOTS client attempts to
       discover DOTS server name(s) using DNS service discovery, as
       specified in Section 8.

   4. Anycast : Send DOTS request to establish a DOTS session with the
       assigned DOTS server anycast address for each combination of
       interface and address family.

   Some of these mechanisms imply the use of DNS to resolve the IP
   address of the DOTS server, while others imply the IP address of the
   relevant DOTS server is obtained directly. Implementation options
   may vary on a per device basis, as some devices may not have DNS
   capabilities and/or proper configuration.

   Clients will prefer information received from the discovery methods
   in the order listed.

   On hosts with more than one interface or address family (IPv4/v6),
   the DOTS server discovery procedure has to be performed for each
   combination of interface and address family. A client MAY choose to
   perform the discovery procedure only for a desired interface/address
   combination if the client does not wish to discover a DOTS server for
   all combinations of interface and address family.

   The above procedure MUST also be followed by a DOTS gateway.

6. Resolution

   Once the DOTS client has retrieved client's DNS domain or discovered
   the DOTS server name that needs to be resolved, an S-NAPTR lookup
   with 'DOTS' application service and the desired protocol tag is made
   to obtain information necessary to connect to the authoritative DOTS
   server within the given domain.

   This specification defines "DOTS" as an application service tag
   (Section 12.3.1) and "signal.udp" (Section 12.3.2), "signal.tcp"
   (Section 12.3.3), and "data.tcp" (Section 12.3.4) as application
   protocol tags.

In the example below, for domain 'example.net', the resolution
algorithm will result in IP address(es), port, tag and protocol tuples as
follows:

```
example.net.
IN NAPTR 100 10 "" DOTS:signal.udp "" signal.example.net.
IN NAPTR 200 10 "" DOTS:signal.tcp "" signal.example.net.
IN NAPTR 300 10 "" DOTS:data.tcp "" data.example.net.

signal.example.net.
IN NAPTR 100 10 S DOTS:signal.udp "" _dots._signal._udp.example.net.
IN NAPTR 200 10 S DOTS:signal.tcp "" _dots._signal._tcp.example.net.

data.example.net.
IN NAPTR 100 10 S DOTS:data.tcp "" _dots._data._tcp.example.net.

_dots._signal._udp.example.net.
IN SRV   0 0 5000 a.example.net.

_dots._signal._tcp.example.net.
IN SRV   0 0 5001 a.example.net.

_dots._data._tcp.example.net.
IN SRV   0 0 5002 a.example.net.

a.example.net.
IN AAAA  2001:db8::1
```

Order	Protocol	IP address	Port	Tag
1	UDP	2001:db8::1	5000	Signal
2	TCP	2001:db8::1	5001	Signal
3	TCP	2001:db8::1	5002	Data

If no DOTS-specific S-NAPTR records can be retrieved, the discovery
procedure fails for this domain name (and the corresponding interface
and IP protocol version). If more domain names are known, the
discovery procedure MAY perform the corresponding S-NAPTR lookups
immediately. However, before retrying a lookup that has failed, a
DOTS client MUST wait a time period that is appropriate for the
encountered error (e.g., NXDOMAIN, timeout, etc.).

7. Discovery using Service Resolution

   This mechanism is performed in two steps:

   1. A DNS domain name is retrieved for each combination of interface
       and address family.

   2. Retrieved DNS domain names are then used for S-NAPTR lookups.
       Further DNS lookups may be necessary to determine DOTS server IP
       address(es).

7.1. Retrieving Domain Name

   A DOTS client has to determine the domain in which it is located.
   The following section describes the means to obtain the domain name
   from DHCP. Other means of retrieving domain names may be used, which
   are outside the scope of this document, e.g., local configuration.

   Implementations MAY allow the user to specify a default name that is
   used, if no specific name has been configured.

7.1.1. DHCP

   DHCP can be used to determine the domain name related to an
   interface's point of network attachment. Network operators may
   provide the domain name to be used for service discovery within an
   access network using DHCP. Sections 3.2 and 3.3 of [RFC5986] define
   DHCP IPv4 and IPv6 access network domain name options,
   OPTION_V4_ACCESS_DOMAIN and OPTION_V6_ACCESS_DOMAIN respectively, to
   identify a domain name that is suitable for service discovery within
   the access network.

   For IPv4, the discovery procedure MUST request the access network
   domain name option in a Parameter Request List option, as described
   in [RFC2131]. [RFC2132] defines the DHCP IPv4 domain name option;
   while this option is less suitable, a client MAY request for it if
   the access network domain name defined in [RFC5986] is not available.

   For IPv6, the discovery procedure MUST request for the access network
   domain name option in an Options Request Option (ORO) within an
   Information-request message, as described in [RFC3315].

   If neither option can be retrieved the procedure fails for this
   interface. If a result can be retrieved it will be used as an input
   for S-NAPTR resolution discussed in Section 6.

8. DNS Service Discovery

   DNS-based Service Discovery (DNS-SD) [RFC6763] and Multicast DNS
   (mDNS) [RFC6762] provide generic solutions for discovering services.
   DNS-SD/mDNS define a set of naming rules for certain DNS record types
   that they use for advertising and discovering services.

8.1. DNS-SD

   Section 4.1 of [RFC6763] specifies that a service instance name in
   DNS-SD has the following structure:

   <Instance> . <Service> . <Domain>

   The <Domain> portion specifies the DNS sub-domain where the service
   instance is registered. It may be "local.", indicating the mDNS
   local domain, or it may be a conventional domain name such as
   "example.com.".

   The <Service> portion of the DOTS service instance name MUST be
   "_dots._signal._udp" or "_dots._signal._tcp" or "_dots._data._tcp".

8.2. mDNS

   A DOTS client can proactively discover DOTS servers being advertised
   in the site by multicasting a PTR query to one or all of the
   following:

   o "_dots._signal._udp.local."

   o "_dots._signal._tcp.local."

   o "_dots._data._tcp.local."

   A DOTS server can send out gratuitous multicast DNS answer packets
   whenever it starts up, wakes from sleep, or detects a change in
   network configuration. DOTS clients receive these gratuitous packets
   and cache information contained in it.

9. DHCP Options for DOTS

   As reported in Section 1.7.2 of [RFC6125]:

      "few certification authorities issue server certificates based on
      IP addresses, but preliminary evidence indicates that such
      certificates are a very small percentage (less than 1%) of issued
      certificates".

In order to allow for PKIX-based authentication between a DOTS client
and server while accommodating for the current best practices for
issuing certificates, this document allows for configuring names to
DOTS clients. These names can be used for two purposes: to retrieve
the list of IP addresses of a DOTS server or to be presented as a
reference identifier for authentication purposes.

Defining the option to include a list of IP addresses would avoid a
dependency on an underlying name resolution, but that design requires
to also supply a name for PKIX-based authentication purposes.

9.1. DHCPv6 DOTS Options

9.1.1. Format of DOTS Reference Identifier Option

The DHCPv6 DOTS option is used to configure a name of the DOTS
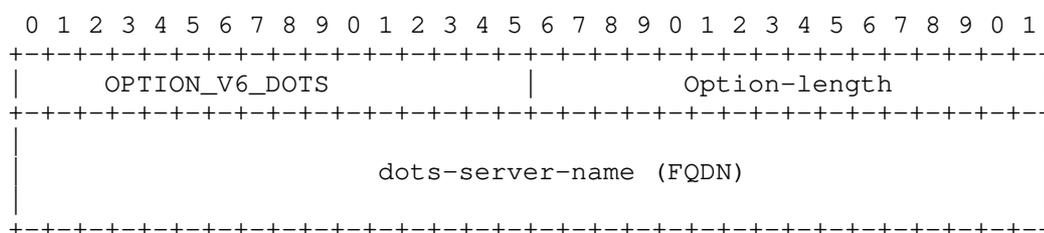server. The format of this option is shown in Figure 2.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_V6_DOTS         |          Option-length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                    dots-server-name (FQDN)                    |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: DHCPv6 DOTS Reference Identifier option

The fields of the option shown in Figure 2 are as follows:

o Option-code: OPTION_V6_DOTS_RI (TBA1, see Section 12.1)
o Option-length: Length of the dots-server-name field in octets.
o dots-server-name: A fully qualified domain name of the DOTS
   server. This field is formatted as specified in Section 8 of
   [RFC3315].

An example of the dots-server-name encoding is shown in Figure 3.
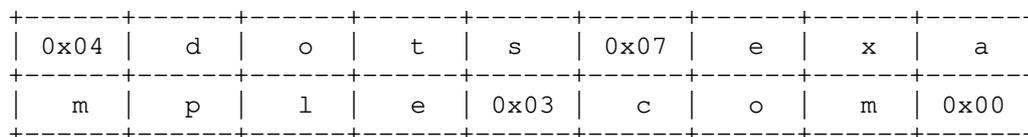This example conveys the FQDN "dots.example.com.".

```
+------+------+------+------+------+------+------+------+------+
| 0x04 |  d   |  o   |  t   |  s   | 0x07 |  e   |  x   |  a   |
+------+------+------+------+------+------+------+------+------+
|  m   |  p   |  l   |  e   | 0x03 |  c   |  o   |  m   | 0x00 |
+------+------+------+------+------+------+------+------+------+
```

Figure 3: An example of the dots-server-name encoding

9.1.2. Format Format of DOTS Address Option

   The DHCPv6 DOTS option can be used to configure a list of IPv6
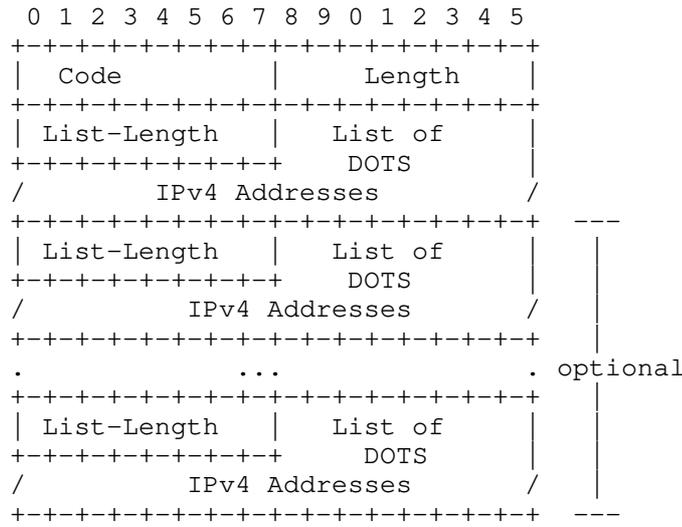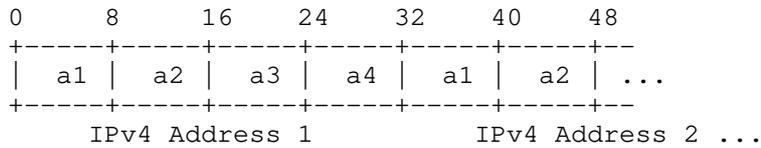   addresses of a DOTS server. The format of this option is shown in
   Figure 4.

```
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |         OPTION_V6_DOTS         |         Option-length         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       |                                                               |
       |                      DOTS ipv6-address                        |
       |                                                               |
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       |                                                               |
       |                      DOTS ipv6-address                        |
       |                                                               |
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                              ...                              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                   Figure 4: DHCPv6 DOTS Address option

   The fields of the option shown in Figure 4 are as follows:

   o Option-code: OPTION_V6_DOTS_ADDRESS (TBA2, see Section 12.1)
   o Option-length: Length of the 'DOTS ipv6-address(es)' field in
      octets. MUST be a multiple of 16.
   o DOTS ipv6-address: Includes one or more IPv6 addresses [RFC4291]
      of the DOTS server to be used by the DOTS client.

      Note, IPv4-mapped IPv6 addresses (Section 2.5.5.2 of [RFC4291])
      are allowed to be included in this option.

   To return more than one DOTS servers to the requesting DHCPv6 client,
   the DHCPv6 server returns multiple instances of OPTION_V6_DOTS.

9.1.3. DHCPv6 Client Behavior

   DHCP clients MAY request options OPTION_V6_DOTS_RI and
   OPTION_V6_DOTS_ADDRESS, as defined in [RFC3315], Sections 17.1.1,
   18.1.1, 18.1.3, 18.1.4, 18.1.5, and 22.7. As a convenience to the
   reader, it is mentioned here that the DHCP client includes the
   requested option codes in the Option Request Option.

If the DHCP client receives more than one instance of
OPTION_V6_DOTS_RI (resp. OPTION_V6_DOTS_ADDRESS) option, it MUST use
only the first instance of that option.

If the DHCP client receives both OPTION_V6_DOTS_RI and
OPTION_V6_DOTS_ADDRESS, the content of OPTION_V6_DOTS_RI is used as
reference identifier for authentication purposes (e.g., PKIX
[RFC6125]), while the addresses included in OPTION_V6_DOTS_ADDRESS
are used to reach the DOTS server. In other words, the name conveyed
in OPTION_V6_DOTS_RI MUST NOT be passed to underlying resolution
library in the presence of OPTION_V6_DOTS_ADDRESS in a response.

If the DHCP client receives OPTION_V6_DOTS_RI only, but
OPTION_V6_DOTS_RI option contains more than one name, as
distinguished by the presence of multiple root labels, the DHCP
client MUST use only the first name. Once the name is validated
(Section 8 of [RFC3315]), the name is passed to a name resolution
library. Moreover, that name is also used as a reference identifier
for authentication purposes.

If the DHCP client receives OPTION_V6_DOTS_ADDRESS only, the
address(es) included in OPTION_V6_DOTS_ADDRESS is used to reach the
DOTS server. In addition, these addresses can be used as identifiers
for authentication.

9.2. DHCPv4 DOTS Options

9.2.1. Format of DOTS Reference Identifier Option

The DHCPv4 DOTS option is used to configure a name of the DOTS
server. The format of this option is illustrated in Figure 5.

```
        Code  Length   DOTS server name
      +-----+-----+-----+-----+-----+-----+-----+--
      | TBA |  n  |  s1 |  s2 |  s3 |  s4 |  s5 |  ...
      +-----+-----+-----+-----+-----+-----+-----+--
```

   The values s1, s2, s3, etc. represent the domain name labels in the
   domain name encoding.

          Figure 5: DHCPv4 DOTS Reference Identifier option

The fields of the option shown in Figure 5 are as follows:

o Code: OPTION_V4_DOTS_RI (TBA3, see Section 12.2);
o Length: Includes the length of the "DOTS server name" field in
   octets; the maximum length is 255 octets.

      o DOTS server name: The domain name of the DOTS server. This field
         is formatted as specified in Section 8 of [RFC3315].

9.2.2. Format Format of DOTS Address Option

   The DHCPv4 DOTS option can be used to configure a list of IPv4
   addresses of a DOTS server. The format of this option is illustrated
   in Figure 6.

```
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  Code         |    Length     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | List-Length   |   List of     |
      +-+-+-+-+-+-+-+   DOTS          |
      /         IPv4 Addresses        /
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---
      | List-Length   |   List of     |   |
      +-+-+-+-+-+-+-+-+   DOTS         |   |
      /          IPv4 Addresses       /   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
      .              ...             . optional
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
      | List-Length   |   List of     |   |
      +-+-+-+-+-+-+-+-+   DOTS         |   |
      /          IPv4 Addresses       /   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---
```

                 Figure 6: DHCPv4 DOTS Address option

   The fields of the option shown in Figure 6 are as follows:

   o Code: OPTION_V4_DOTS_ADDRESS (TBA4, see Section 12.2);
   o Length: Length of all included data in octets. The minimum length
      is 5.
   o List-Length: Length of the "List of DOTS IPv4 Addresses" field in
      octets; MUST be a multiple of 4.
   o List of DOTS IPv4 Addresses: Contains one or more IPv4 addresses
      of the DOTS server to be used by the DOTS client. The format of
      this field is shown in Figure 7.
   o OPTION_V4_DOTS can include multiple lists of DOTS IPv4 addresses;
      each list is treated separately as it corresponds to a given DOTS
      server.

      When several lists of DOTS IPv4 addresses are to be included,
      "List-Length" and "DOTS IPv4 Addresses" fields are repeated.

```
         0     8    16    24    32    40    48
         +-----+-----+-----+-----+-----+-----+--
         | a1  | a2  | a3  | a4  | a1  | a2  | ...
         +-----+-----+-----+-----+-----+-----+--
              IPv4 Address 1         IPv4 Address 2 ...
```

   This format assumes that an IPv4 address is encoded as a1.a2.a3.a4.

             Figure 7: Format of the List of DOTS IPv4 Addresses

   OPTION_V4_DOTS is a concatenation-requiring option. As such, the
   mechanism specified in [RFC3396] MUST be used if OPTION_V4_DOTS
   exceeds the maximum DHCPv4 option size of 255 octets.

9.2.3. DHCPv4 Client Behavior

   To discover a DOTS server, the DHCPv4 client MUST include both
   OPTION_V4_DOTS_RI and OPTION_V4_DOTS_ADDRESS in a Parameter Request
   List Option [RFC2132].

   If the DHCP client receives more than one instance of
   OPTION_V4_DOTS_RI (resp. OPTION_V4_DOTS_ADDRESS) option, it MUST use
   only the first instance of that option.

   If the DHCP client receives both OPTION_V4_DOTS_RI and
   OPTION_V4_DOTS_ADDRESS, the content of OPTION_V6_DOTS_RI is used as
   reference identifier for authentication purposes, while the addresses
   included in OPTION_V4_DOTS_ADDRESS are used to reach the DOTS server.
   In other words, the name conveyed in OPTION_V4_DOTS_RI MUST NOT be
   passed to underlying resolution library in the presence of
   OPTION_V4_DOTS_ADDRESS in a response.

   If the DHCP client receives OPTION_V4_DOTS_RI only, but
   OPTION_V4_DOTS_RI option contains more than one name, as
   distinguished by the presence of multiple root labels, the DHCP
   client MUST use only the first name. Once the name is validated
   (Section 8 of [RFC3315]), the name is passed to a name resolution
   library. Moreover, that name is also used as a reference identifier
   for authentication purposes.

   If the DHCP client receives OPTION_V4_DOTS_ADDRESS only, the
   address(es) included in OPTION_V4_DOTS_ADDRESS is used to reach the
   DOTS server. In addition, these addresses can be used as identifiers
   for authentication.

10. Anycast

   IP anycast can also be used for DOTS service discovery. A packet
   sent to an anycast address is delivered to the 'topologically
   nearest' network interface with the anycast address.

   When a DOTS client requires DOTS services, it attempts to establish a
   signaling session with the assigned anycast address(es) defined in
   Sections 12.4 and 12.5. A DOTS server, that receives a DOTS request
   with an anycast address, SHOULD redirect the DOTS client to the
   appropriate DOTS unicast server(s) using the mechanism described in
   Section 5.5 of [I-D.ietf-dots-signal-channel], unless it is
   configured otherwise. Indeed, a DOTS server SHOULD be configurable
   to maintain all DOTS communications using anycast. DOTS redirect is
   not made mandatory because the use of anycast is not problematic for
   some deployment scenarios such as an enterprise network deploying one
   single DOTS gateway connected to one single network provider.

   [I-D.boucadair-dots-multihoming] identifies a set of deployment
   schemes in which the use of anycast is not recommended.

11. Security Considerations

   DOTS-related security considerations are discussed in Section 4 of
   [I-D.ietf-dots-architecture] is to be considered. DOTS agents must
   authenticate each other using (D)TLS before a DOTS session is
   considered valid.

   If the DOTS client is explicitly configured with DOTS server(s) then
   the DOTS client can also be explicitly configured with credentials to
   authenticate the DOTS server.

   The CPE device acting as a DOTS client MAY use Bootstrapping Remote
   Secure Key Infrastructures (BRSKI) discussed in
   [I-D.ietf-anima-bootstrapping-keyinfra] to automatically bootstrap
   using the vendor installed X.509 certificate, in combination with a
   domain registrar provided by the upstream transit provider and
   vendor's authorizing service. The CPE device authenticates to the
   upstream transit provider using the vendor installed X.509
   certificate and the upstream transit provider validates the vendor
   installed certificate on the CPE device using the Manufacturer
   Authorized Signing Authority (MASA) service. If authentication is
   successful then the CPE device can request and get a voucher from the
   MASA service via the domain registrar. The voucher is signed by the
   MASA service and includes the upstream transit provider's trust
   anchor certificate. The CPE device validates the signed voucher
   using the manufacturer installed trust anchor associated with the
   vendor's selected MASA service and stores the upstream transit

provider's trust anchor certificate. The CPE device then uses
Enrollment over Secure Transport (EST) [RFC7030] for certificate
enrollment (Section 3.8 in [I-D.ietf-anima-bootstrapping-keyinfra]).
The DOTS client on the CPE device can authenticate to the DOTS server
using the certificate provisioned by the EST server and the DOTS
client can validate the DOTS server certificate using the upstream
transit provider's trust anchor certificate it had received in the
voucher.

11.1. DHCP

The security considerations in [RFC2131] and [RFC3315] are to be
considered.

11.2. Service Resolution

The primary attack against the methods described in Section 7 is one
that would lead to impersonation of a DOTS server. An attacker could
attempt to compromise the S-NAPTR resolution. The use of mutual
authentication makes it difficult to redirect a DOTS client to an
illegitimate DOTS server.

11.3. DNS Service Discovery

Since DNS-SD is just a specification for how to name and use records
in the existing DNS system, it has no specific additional security
requirements over and above those that already apply to DNS queries
and DNS updates. For DNS queries, DNS Security Extensions (DNSSEC)
[RFC4033] SHOULD be used where the authenticity of information is
important. For DNS updates, secure updates [RFC2136][RFC3007] SHOULD
generally be used to control which clients have permission to update
DNS records.

For mDNS, in addition to what has been described above, a principal
security threat is a security threat inherent to IP multicast routing
and any application that runs on it. A rogue system can advertise
that it is a DOTS server. Discovery of such rogue systems as DOTS
servers, in itself, is not a security threat if the DOTS client
authenticates the discovered DOTS servers.

11.4. Anycast

Anycast-related security considerations are discussed in [RFC4786]
and [RFC7094].

12. IANA Considerations

   IANA is requested to allocate the SRV service name of "_dots._signal"
   for DOTS signal channel over UDP or TCP, and the service name of
   "_dots._data" for DOTS data channel over TCP.

12.1. DHCPv6 Option

   IANA is requested to assign the following new DHCPv6 Option Code in
   the registry maintained in http://www.iana.org/assignments/
   dhcpv6-parameters:

```
                     Option Name Value
               --------------------- -----
                   OPTION_V6_DOTS_RI TBA1
              OPTION_V6_DOTS_ADDRESS TBA2
```

12.2. DHCPv4 Option

   IANA is requested to assign the following new DHCPv4 Option Code in
   the registry maintained in http://www.iana.org/assignments/bootp-
   dhcp-parameters/:

Option Name	Value	Data length	Meaning
OPTION_V4_DOTS_RI	TBA3	Variable; the maximum length is 255 octets.	Includes the name of the DOTS server.
OPTION_V4_DOTS_ADDRESS	TBA4	Variable; the minimum length is 5.	Includes one or multiple lists of DOTS IP addresses; each list is treated as a separate DOTS server.

12.3. Application Service & Application Protocol Tags

   This document requests IANA to make the following allocations from
   the registry available at: https://www.iana.org/assignments/s-naptr-
   parameters/s-naptr-parameters.xhtml.

12.3.1. DOTS Application Service Tag Registration

   o Application Protocol Tag: DOTS

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.2. signal.udp Application Protocol Tag Registration

   o Application Protocol Tag: signal.udp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.3. signal.tcp Application Protocol Tag Registration

   o Application Protocol Tag: signal.tcp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.3.4. data.tcp Application Protocol Tag Registration

   o Application Protocol Tag: data.tcp

   o Intended Usage: See Section 6

   o Security Considerations: See Section 11

   o Contact Information: <one of the authors>

12.4. IPv4 Anycast

   IANA has assigned a single IPv4 address from the 192.0.0.0/24 prefix
   and registered it in the "IANA IPv4 Special-Purpose Address Registry"
   [RFC6890].

```
+--------------------+----------------------------------------+
| Attribute          | Value                                  |
+--------------------+----------------------------------------+
| Address Block      | TBA                                    |
| Name               | Distributed-Denial-of-Service Open Threat |
|                    | Signaling (DOTS) Anycast               |
| RFC                | <this document>                        |
| Allocation Date    | <date of approval of this document>    |
| Termination Date   | N/A                                    |
| Source             | True                                   |
| Destination        | True                                   |
| Forwardable        | True                                   |
| Global             | True                                   |
| Reserved-by-Protocol | False                                |
+--------------------+----------------------------------------+
```

12.5. IPv6 Anycast

   IANA has assigned a single IPv6 address from the 2001:0000::/23
   prefix and registered it in the "IANA IPv6 Special-Purpose Address
   Registry" [RFC6890].

```
+--------------------+----------------------------------------+
| Attribute          | Value                                  |
+--------------------+----------------------------------------+
| Address Block      | TBA                                    |
| Name               | Distributed-Denial-of-Service Open Threat |
|                    | Signaling (DOTS) Anycast               |
| RFC                | <this document>                        |
| Allocation Date    | <date of approval of this document>    |
| Termination Date   | N/A                                    |
| Source             | True                                   |
| Destination        | True                                   |
| Forwardable        | True                                   |
| Global             | True                                   |
| Reserved-by-Protocol | False                                |
+--------------------+----------------------------------------+
```

13. Acknowledgements

   Thanks to Brian Carpenter for the review of the BRSKI text.

   Many thanks to Russ White for the review, comments, and text
   contribution.

14. References

14.1. Normative References

   [I-D.ietf-dots-architecture]
            Mortensen, A., Andreasen, F., K, R.,
            christopher_gray3@cable.comcast.com, c., Compton, R., and
            N. Teague, "Distributed-Denial-of-Service Open Threat
            Signaling (DOTS) Architecture", draft-ietf-dots-
            architecture-07 (work in progress), September 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>.

   [RFC2131] Droms, R., "Dynamic Host Configuration Protocol",
            RFC 2131, DOI 10.17487/RFC2131, March 1997,
            <https://www.rfc-editor.org/info/rfc2131>.

   [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor
            Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,
            <https://www.rfc-editor.org/info/rfc2132>.

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
            C., and M. Carney, "Dynamic Host Configuration Protocol
            for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
            2003, <https://www.rfc-editor.org/info/rfc3315>.

   [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the
            Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396,
            DOI 10.17487/RFC3396, November 2002,
            <https://www.rfc-editor.org/info/rfc3396>.

   [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application
            Service Location Using SRV RRs and the Dynamic Delegation
            Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958,
            January 2005, <https://www.rfc-editor.org/info/rfc3958>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
            Architecture", RFC 4291, DOI 10.17487/RFC4291, February
            2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local
            Location Information Server (LIS)", RFC 5986,
            DOI 10.17487/RFC5986, September 2010,
            <https://www.rfc-editor.org/info/rfc5986>.

   [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,
              DOI 10.17487/RFC6762, February 2013,
              <https://www.rfc-editor.org/info/rfc6762>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [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>.

   [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>.

14.2. Informative References

   [I-D.boucadair-dots-multihoming]
              Boucadair, M. and R. K, "Multi-homing Deployment
              Considerations for Distributed-Denial-of-Service Open
              Threat Signaling (DOTS)", draft-boucadair-dots-
              multihoming-03 (work in progress), April 2018.

   [I-D.ietf-anima-bootstrapping-keyinfra]
              Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
              S., and K. Watsen, "Bootstrapping Remote Secure Key
              Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
              keyinfra-16 (work in progress), June 2018.

   [I-D.ietf-dots-signal-channel]
              K, R., Boucadair, M., Patil, P., Mortensen, A., and N.
              Teague, "Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Signal Channel Specification", draft-
              ietf-dots-signal-channel-25 (work in progress), September
              2018.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-16 (work
              in progress), July 2018.

   [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>.

   [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>.

   [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>.

   [RFC4732]   Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
               Denial-of-Service Considerations", RFC 4732,
               DOI 10.17487/RFC4732, December 2006,
               <https://www.rfc-editor.org/info/rfc4732>.

   [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>.

   [RFC6125]   Saint-Andre, P. and J. Hodges, "Representation and
               Verification of Domain-Based Application Service Identity
               within Internet Public Key Infrastructure Using X.509
               (PKIX) Certificates in the Context of Transport Layer
               Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
               2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC7030]   Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
               "Enrollment over Secure Transport", RFC 7030,
               DOI 10.17487/RFC7030, October 2013,
               <https://www.rfc-editor.org/info/rfc7030>.

   [RFC7094]   McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
               "Architectural Considerations of IP Anycast", RFC 7094,
               DOI 10.17487/RFC7094, January 2014,
               <https://www.rfc-editor.org/info/rfc7094>.

Authors' Addresses

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

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

Email: TirumaleswarReddy_Konda@McAfee.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

        Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
                    draft-ietf-dots-architecture-05

Abstract

   This document describes an architecture for establishing and
   maintaining Distributed Denial of Service (DDoS) Open Threat
   Signaling (DOTS) within and between domains. The document does not
   specify protocols or protocol extensions, instead focusing on
   defining architectural relationships, components and concepts used in
   a DOTS deployment.

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 April 28, 2018.

Copyright Notice

Table of Contents

1. Context and Motivation

   Signaling the need for help defending against an active distributed
   denial of service (DDoS) attack requires a common understanding of
   mechanisms and roles among the parties coordinating defensive
   response. The signaling layer and supplementary messaging is the
   focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of
   coordinating defensive measures among willing peers to mitigate
   attacks quickly and efficiently, enabling hybrid attack responses
   coordinated locally at or near the target of an active attack, or
   anywhere in-path between attack sources and target. Sample DOTS use
   cases are elaborated in [I-D.ietf-dots-use-cases].

   This document describes an architecture used in establishing,
   maintaining or terminating a DOTS relationship within a domain or
   between domains.

1.1. Terminology

1.1.1. Key Words

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

1.1.2. Definition of Terms

   This document uses the terms defined in [I-D.ietf-dots-requirements].

1.2. Scope

   In this architecture, DOTS clients and servers communicate using DOTS
   signaling. As a result of signals from a DOTS client, the DOTS
   server may modify the forwarding path of traffic destined for the
   attack target(s), for example by diverting traffic to a mitigator or
   pool of mitigators, where policy may be applied to distinguish and
   discard attack traffic. Any such policy is deployment-specific.

   The DOTS architecture presented here is applicable across network
   administrative domains - for example, between an enterprise domain
   and the domain of a third-party attack mitigation service - as well
   as to a single administrative domain. DOTS is generally assumed to
   be most effective when aiding coordination of attack response between

two or more participating networks, but single domain scenarios are
valuable in their own right, as when aggregating intra-domain DOTS
client signals for inter-domain coordinated attack response.

This document does not address any administrative or business
agreements that may be established between involved DOTS parties.
Those considerations are out of scope. Regardless, this document
assumes necessary authentication and authorization mechanisms are put
in place so that only authorized clients can invoke the DOTS service.

A detailed set of DOTS requirements are discussed in
[I-D.ietf-dots-requirements], and the DOTS architecture is designed
to follow those requirements. Only new behavioral requirements are
described in this document.

1.3. Assumptions

This document makes the following assumptions:

o All domains in which DOTS is deployed are assumed to offer the
   required connectivity between DOTS agents and any intermediary
   network elements, but the architecture imposes no additional
   limitations on the form of connectivity.

o Congestion and resource exhaustion are intended outcomes of a DDoS
   attack [RFC4732]. Some operators may utilize non-impacted paths
   or networks for DOTS, but in general conditions should be assumed
   to be hostile and DOTS must be able to function in all
   circumstances, including when the signaling path is significantly
   impaired.

o There is no universal DDoS attack scale threshold triggering a
   coordinated response across administrative domains. A network
   domain administrator, or service or application owner may
   arbitrarily set attack scale threshold triggers, or manually send
   requests for mitigation.

o Mitigation requests may be sent to one or more upstream DOTS
   servers based on criteria determined by DOTS client administrators
   and the underlying network configuration. The number of DOTS
   servers with which a given DOTS client has established
   communications is determined by local policy and is deployment-
   specific. For example, a DOTS client of a multi-homed network may
   support built-in policies to establish DOTS relationships with
   DOTS servers located upstream of each interconnection link.

o The mitigation capacity and/or capability of domains receiving
   requests for coordinated attack response is opaque to the domains

sending the request. The domain receiving the DOTS client signal
may or may not have sufficient capacity or capability to filter
any or all DDoS attack traffic directed at a target. In either
case, the upstream DOTS server may redirect a request to another
DOTS server. Redirection may be local to the redirecting DOTS
server's domain, or may involve a third-party domain.

o DOTS client and server signals, as well as messages sent through
   the data channel, are sent across any transit networks with the
   same probability of delivery as any other traffic between the DOTS
   client domain and the DOTS server domain. Any encapsulation
   required for successful delivery is left untouched by transit
   network elements. DOTS server and DOTS client cannot assume any
   preferential treatment of DOTS signals. Such preferential
   treatment may be available in some deployments (e.g., intra-domain
   scenarios), and the DOTS architecture does not preclude its use
   when available. However, DOTS itself does not address how that
   may be done.

o The architecture allows for, but does not assume, the presence of
   Quality of Service (QoS) policy agreements between DOTS-enabled
   peer networks or local QoS prioritization aimed at ensuring
   delivery of DOTS messages between DOTS agents. QoS is an
   operational consideration only, not a functional part of the DOTS
   architecture.

o The signal and data channels are loosely coupled, and may not
   terminate on the same DOTS server.

2. DOTS Architecture

   The basic high-level DOTS architecture is illustrated in Figure 1:

```
   +-----------+                  +-------------+
   | Mitigator | ~~~~~~~~~~       | DOTS Server |
   +-----------+                  +-------------+
                                         |
                                         |
                                         |
   +---------------+      +-------------+
   | Attack Target | ~~~~~| DOTS Client |
   +---------------+      +-------------+
```

                    Figure 1: Basic DOTS Architecture

   A simple example instantiation of the DOTS architecture could be an
   enterprise as the attack target for a volumetric DDoS attack, and an
   upstream DDoS mitigation service as the mitigator. The enterprise

(attack target) is connected to the Internet via a link that is
getting saturated, and the enterprise suspects it is under DDoS
attack. The enterprise has a DOTS client, which obtains information
about the DDoS attack, and signals the DOTS server for help in
mitigating the attack. The DOTS server in turn invokes one or more
mitigators, which are tasked with mitigating the actual DDoS attack,
and hence aim to suppress the attack traffic while allowing valid
traffic to reach the attack target.

The scope of the DOTS specifications is the interfaces between the
DOTS client and DOTS server. The interfaces to the attack target and
the mitigator are out of scope of DOTS. Similarly, the operation of
both the attack target and the mitigator is out of scope of DOTS.
Thus, DOTS neither specifies how an attack target decides it is under
DDoS attack, nor does DOTS specify how a mitigator may actually
mitigate such an attack. A DOTS client's request for mitigation is
advisory in nature, and may not lead to any mitigation at all,
depending on the DOTS server domain's capacity and willingness to
mitigate on behalf of the DOTS client's domain.

The DOTS client may be provided with a list of DOTS servers, each
associated with one or more IP addresses. These addresses may or may
not be of the same address family. The DOTS client establishes one
or more sessions by connecting to the provided DOTS server addresses.

As illustrated in Figure 2, there are two interfaces between a DOTS
server and a DOTS client; a signal channel and (optionally) a data
channel.

```
+---------------+                                +---------------+
|               | <------- Signal Channel ------> |               |
|  DOTS Client  |                                |  DOTS Server  |
|               | <=======  Data Channel  ======> |               |
+---------------+                                +---------------+
```

Figure 2: DOTS Interfaces

The primary purpose of the signal channel is for a DOTS client to ask
a DOTS server for help in mitigating an attack, and for the DOTS
server to inform the DOTS client about the status of such mitigation.
The DOTS client does this by sending a client signal, which contains
information about the attack target(s). The client signal may also
include telemetry information about the attack, if the DOTS client
has such information available. The DOTS server in turn sends a
server signal to inform the DOTS client of whether it will honor the
mitigation request. Assuming it will, the DOTS server initiates
attack mitigation, and periodically informs the DOTS client about the
status of the mitigation. Similarly, the DOTS client periodically

informs the DOTS server about the client's status, which at a minimum
provides client (attack target) health information, but it should
also include efficacy information about the attack mitigation as it
is now seen by the client. At some point, the DOTS client may decide
to terminate the server-side attack mitigation, which it indicates to
the DOTS server over the signal channel. A mitigation may also be
terminated if a DOTS client-specified mitigation lifetime is
exceeded. Note that the signal channel may need to operate over a
link that is experiencing a DDoS attack and hence is subject to
severe packet loss and high latency.

While DOTS is able to request mitigation with just the signal
channel, the addition of the DOTS data channel provides for
additional and more efficient capabilities. The primary purpose of
the data channel is to support DOTS related configuration and policy
information exchange between the DOTS client and the DOTS server.
Examples of such information include, but are not limited to:

o Creating identifiers, such as names or aliases, for resources for
   which mitigation may be requested. Such identifiers may then be
   used in subsequent signal channel exchanges to refer more
   efficiently to the resources under attack, as seen in Figure 3,
   using JSON to serialize the data:

```
{
    "https1": [
        "192.0.2.1:443",
        "198.51.100.2:443",
    ],
    "proxies": [
        "203.0.113.3:3128",
        "[2001:db8:ac10::1]:3128"
    ],
    "api_urls": "https://apiserver.example.com/api/v1",
}
```

               Figure 3: Protected resource identifiers

o Black-list management, which enables a DOTS client to inform the
   DOTS server about sources to suppress.

o White-list management, which enables a DOTS client to inform the
   DOTS server about sources from which traffic is always accepted.

o Filter management, which enables a DOTS client to install or
   remove traffic filters dropping or rate-limiting unwanted traffic.

o DOTS client provisioning.

Note that while it is possible to exchange the above information
before, during or after a DDoS attack, DOTS requires reliable
delivery of this information and does not provide any special means
for ensuring timely delivery of it during an attack. In practice,
this means that DOTS deployments should not rely on such information
being exchanged during a DDoS attack.

2.1. DOTS Operations

DOTS does not prescribe any specific deployment models, however DOTS
is designed with some specific requirements around the different DOTS
agents and their relationships.

First of all, a DOTS agent belongs to a domain that has an identity
which can be authenticated and authorized. DOTS agents communicate
with each other over a mutually authenticated signal channel and
(optionally) data channel. However, before they can do so, a service
relationship needs to be established between them. The details and
means by which this is done is outside the scope of DOTS, however an
example would be for an enterprise A (DOTS client) to sign up for
DDoS service from provider B (DOTS server). This would establish a
(service) relationship between the two that enables enterprise A's
DOTS client to establish a signal channel with provider B's DOTS
server. A and B will authenticate each other, and B can verify that
A is authorized for its service.

From an operational and design point of view, DOTS assumes that the
above relationship is established prior to a request for DDoS attack
mitigation. In particular, it is assumed that bi-directional
communication is possible at this time between the DOTS client and
DOTS server. Furthermore, it is assumed that additional service
provisioning, configuration and information exchange can be performed
by use of the data channel, if operationally required. It is not
until this point that the mitigation service is available for use.

Once the mutually authenticated signal channel has been established,
it will remain active. This is done to increase the likelihood that
the DOTS client can signal the DOTS server for help when the attack
target is being flooded, and similarly raise the probability that
DOTS server signals reach the client regardless of inbound link
congestion. This does not necessarily imply that the attack target
and the DOTS client have to be co-located in the same administrative
domain, but it is expected to be a common scenario.

DDoS mitigation with the help of an upstream mitigator may involve
some form of traffic redirection whereby traffic destined for the
attack target is steered towards the mitigator. Common mechanisms to
achieve this redirection depend on BGP [RFC4271] and DNS [RFC1035].

The mitigator in turn inspects and scrubs the traffic, and forwards
the resulting (hopefully non-attack) traffic to the attack target.
Thus, when a DOTS server receives an attack mitigation request from a
DOTS client, it can be viewed as a way of causing traffic redirection
for the attack target indicated.

DOTS relies on mutual authentication and the pre-established service
relationship between the DOTS client's domain and the DOTS server's
domain to provide basic authorization. The DOTS server should
enforce additional authorization mechanisms to restrict the
mitigation scope a DOTS client can request, but such authorization
mechanisms are deployment-specific.

Although co-location of DOTS server and mitigator within the same
domain is expected to be a common deployment model, it is assumed
that operators may require alternative models. Nothing in this
document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help
coordinating attack response originate. The requests may be in
response to an active, ongoing attack against a target in the DOTS
client's domain, but no active attack is required for a DOTS client
to request help. Operators may wish to have upstream mitigators in
the network path for an indefinite period, and are restricted only by
business relationships when it comes to duration and scope of
requested mitigation.

The DOTS client requests attack response coordination from a DOTS
server over the signal channel, including in the request the DOTS
client's desired mitigation scoping, as described in
[I-D.ietf-dots-requirements]. The actual mitigation scope and
countermeasures used in response to the attack are up to the DOTS
server and mitigator operators, as the DOTS client may have a narrow
perspective on the ongoing attack. As such, the DOTS client's
request for mitigation should be considered advisory: guarantees of
DOTS server availability or mitigation capacity constitute service
level agreements and are out of scope for this document.

The DOTS client adjusts mitigation scope and provides available
mitigation feedback (e.g. mitigation efficacy) at the direction of
its local administrator. Such direction may involve manual or
automated adjustments in response to updates from the DOTS server.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct session, the DOTS client sends
a heartbeat over the signal channel to maintain its half of the
channel. The DOTS client similarly expects a heartbeat from the DOTS
server, and may consider a session terminated in the extended absence
of a DOTS server heartbeat.

2.2.2. DOTS Server

A DOTS server is a DOTS agent capable of receiving, processing and
possibly acting on requests for help coordinating attack response
from DOTS clients. The DOTS server authenticates and authorizes DOTS
clients as described in Section 3.1, and maintains session state,
tracking requests for mitigation, reporting on the status of active
mitigations, and terminating sessions in the extended absence of a
client heartbeat or when a session times out.

Assuming the preconditions discussed below exist, a DOTS client
maintaining an active session with a DOTS server may reasonably
expect some level of mitigation in response to a request for
coordinated attack response.

The DOTS server enforces authorization of DOTS clients' signals for
mitigation. The mechanism of enforcement is not in scope for this
document, but is expected to restrict requested mitigation scope to
addresses, prefixes, and/or services owned by the DOTS client's
administrative domain, such that a DOTS client from one domain is not
able to influence the network path to another domain. A DOTS server
MUST reject requests for mitigation of resources not owned by the
requesting DOTS client's administrative domain. A DOTS server MAY
also refuse a DOTS client's mitigation request for arbitrary reasons,
within any limits imposed by business or service level agreements
between client and server domains. If a DOTS server refuses a DOTS
client's request for mitigation, the DOTS server SHOULD include the
refusal reason in the server signal sent to the client.

A DOTS server is in regular contact with one or more mitigators. If
a DOTS server accepts a DOTS client's request for help, the DOTS
server forwards a translated form of that request to the mitigator(s)
responsible for scrubbing attack traffic. Note that the form of the
translated request passed from the DOTS server to the mitigator is
not in scope: it may be as simple as an alert to mitigator operators,
or highly automated using vendor or open application programming
interfaces supported by the mitigator. The DOTS server MUST report
the actual scope of any mitigation enabled on behalf of a client.

The DOTS server SHOULD retrieve available metrics for any mitigations
activated on behalf of a DOTS client, and SHOULD include them in

server signals sent to the DOTS client originating the request for
mitigation.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct channel, the DOTS server MUST
send a heartbeat over the signal channel to maintain its half of the
channel. The DOTS server similarly expects a heartbeat from the DOTS
client, and MAY consider a session terminated in the extended absence
of a DOTS client heartbeat.

2.2.3. DOTS Gateway

Traditional client/server relationships may be expanded by chaining
DOTS sessions. This chaining is enabled through "logical
concatenation" of a DOTS server and a DOTS client, resulting in an
application analogous to the Session Initiation Protocol (SIP)
[RFC3261] logical entity of a Back-to-Back User Agent (B2BUA)
[RFC7092]. The term DOTS gateway is used here in the descriptions of
selected scenarios involving this application.

A DOTS gateway may be deployed client-side, server-side or both. The
gateway may terminate multiple discrete client connections and may
aggregate these into a single or multiple DOTS sessions.

The DOTS gateway will appear as a server to its downstream agents and
as a client to its upstream agents, a functional concatenation of the
DOTS client and server roles, as depicted in Figure 4:

```
                       +-------------+
                       |      | D |   |
      +----+           |      | O |   |           +----+
      | c1 |-----------| s1   | T | c2|---------| s2 |
      +----+           |      | S |   |           +----+
                       |      | G |   |
                       +-------------+
```

                      Figure 4: DOTS gateway

The DOTS gateway MUST perform full stack DOTS session termination and
reorigination between its client and server side. The details of how
this is achieved are implementation specific. The DOTS protocol does
not include any special features related to DOTS gateways, and hence
from a DOTS perspective, whenever a DOTS gateway is present, the DOTS
session simply terminates/originates there.

2.3. DOTS Agent Relationships

   So far, we have only considered a relatively simple scenario of a
   single DOTS client associated with a single DOTS server, however DOTS
   supports more advanced relationships.

   A DOTS server may be associated with one or more DOTS clients, and
   those DOTS clients may belong to different domains. An example
   scenario is a mitigation provider serving multiple attack targets
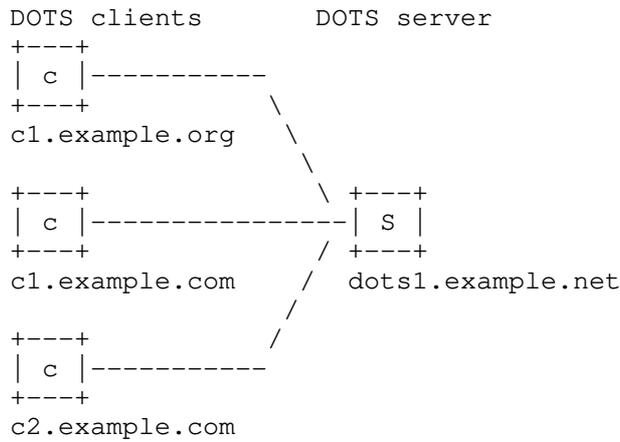   (Figure 5).

```
      DOTS clients        DOTS server
      +---+
      | c |-----------
      +---+            \
      c1.example.org    \
                         \
      +---+               \ +---+
      | c |---------------| S |
      +---+              / +---+
      c1.example.com    /  dots1.example.net
                       /
      +---+           /
      | c |-----------
      +---+
      c2.example.com
```

                 Figure 5: DOTS server with multiple clients

   A DOTS client may be associated with one or more DOTS servers, and
   those DOTS servers may belong to different domains. This may be to
   ensure high availability or co-ordinate mitigation with more than one
   directly connected ISP. An example scenario is for an enterprise to
   have DDoS mitigation service from multiple providers, as shown in
   Figure 6.

```
         DOTS client        DOTS servers
                             +---+
              -----------| s |
                 /           +---+
                /            dots1.example.net
               /
   +---+/            +---+
   | c |-------------| s |
   +---+\            +---+
        \            dots.example.org
          \
            \        +---+
              -----------| s |
                         +---+
         c.example.com   dots2.example.net
```

                  Figure 6: Multi-Homed DOTS Client

   Deploying a multi-homed client requires extra care and planning, as
   the DOTS servers with which the multi-homed client communicates may
   not be affiliated. Should the multi-homed client simultaneously
   request for mitigation from all servers with which it has established
   signal channels, the client may unintentionally inflict additional
   network disruption on the resources it intends to protect. In one of
   the worst cases, a multi-homed DOTS client could cause a permanent
   routing loop of traffic destined for the client's protected services,
   as the uncoordinated DOTS servers' mitigators all try to divert that
   traffic to their own scrubbing centers.

   The DOTS protocol itself provides no fool-proof method to prevent
   such self-inflicted harms as a result deploying multi-homed DOTS
   clients. If DOTS client implementations nevertheless include support
   for multi-homing, they are expected to be aware of the risks, and
   consequently to include measures aimed at reducing the likelihood of
   negative outcomes. Simple measures might include:

   o Requesting mitigation serially, ensuring only one mitigation
      request for a given address space is active at any given time;

   o Dividing the protected resources among the DOTS servers, such that
      no two mitigators will be attempting to divert and scrub the same
      traffic;

   o Restricting multi-homing to deployments in which all DOTS servers
      are coordinating management of a shared pool of mitigation
      resources.

2.3.1. Gatewayed Signaling

   As discussed in Section 2.2.3, a DOTS gateway is a logical function
   chaining DOTS sessions through concatenation of a DOTS server and
   DOTS client.

   An example scenario, as shown in Figure 7 and Figure 8, is for an
   enterprise to have deployed multiple DOTS capable devices which are
   able to signal intra-domain using TCP [RFC0793] on un-congested links
   to a DOTS gateway which may then transform these to a UDP [RFC0768]
   transport inter-domain where connection oriented transports may
   degrade; this applies to the signal channel only, as the data channel
   requires a connection-oriented transport. The relationship between
   the gateway and its upstream agents is opaque to the initial clients.

```
            +---+
            | c |\
            +---+ \              +---+
                   \-----TCP-----| D |                 +---+
            +---+                | O |                 |   |
            | c |--------TCP-----| T |------UDP------| S |
            +---+                | S |                 |   |
                   /-----TCP-----| G |                 +---+
            +---+ /              +---+
            | c |/
            +---+
            example.com          example.com           example.net
            DOTS clients         DOTS gateway (DOTSG)  DOTS server
```

           Figure 7: Client-Side Gateway with Aggregation

```
            +---+
            | c |\
            +---+ \              +---+
                   \-----TCP-----| D |------UDP------+---+
            +---+                | O |               |   |
            | c |--------TCP-----| T |------UDP------| S |
            +---+                | S |               |   |
                   /-----TCP-----| G |------UDP------+---+
            +---+ /              +---+
            | c |/
            +---+
            example.com          example.com           example.net
            DOTS clients         DOTS gateway (DOTSG)  DOTS server
```

          Figure 8: Client-Side Gateway without Aggregation

This may similarly be deployed in the inverse scenario where the
gateway resides in the server-side domain and may be used to
terminate and/or aggregate multiple clients to single transport as
shown in figures Figure 9 and Figure 10.

```
        +---+
        | c |\
        +---+ \              +---+
               \-----UDP-----| D |              +---+
        +---+                | O |              |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+                | S |              |   |
               /-----TCP-----| G |              +---+
        +---+ /              +---+
        | c |/
        +---+
        example.com        example.net          example.net
        DOTS clients       DOTS gateway (DOTSG)  DOTS server
```

              Figure 9: Server-Side Gateway with Aggregation

```
        +---+
        | c |\
        +---+ \              +---+
               \-----UDP-----| D |------TCP------+---+
        +---+                | O |              |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+                | S |              |   |
               /-----UDP-----| G |------TCP------+---+
        +---+ /              +---+
        | c |/
        +---+
        example.com        example.net          example.net
        DOTS clients       DOTS gateway (DOTSG)  DOTS server
```

            Figure 10: Server-Side Gateway without Aggregation

   This document anticipates scenarios involving multiple DOTS gateways.
   An example is a DOTS gateway at the network client's side, and
   another one at the server side. The first gateway can be located at
   a CPE to aggregate requests from multiple DOTS clients enabled in an
   enterprise network. The second DOTS gateway is deployed on the
   provider side. This scenario can be seen as a combination of the
   client-side and server-side scenarios.

3. Concepts

3.1. DOTS Sessions

   In order for DOTS to be effective as a vehicle for DDoS mitigation
   requests, one or more DOTS clients must establish ongoing
   communication with one or more DOTS servers. While the preconditions
   for enabling DOTS in or among network domains may also involve
   business relationships, service level agreements, or other formal or
   informal understandings between network operators, such
   considerations are out of scope for this document.

   A DOTS session is established to support bilateral exchange of data
   between an associated DOTS client and a DOTS server. In the DOTS
   architecture, data is exchanged between DOTS agents over signal and
   data channels. Regardless, a DOTS session is characterized by the
   presence of an established signal channel. A data channel may be
   established as well, however it is not a prerequisite. Conversely, a
   DOTS session cannot exist without an established signal channel: when
   an established signal channel is terminated for any reason, the DOTS
   session is also said to be terminated.

3.1.1. Preconditions

   Prior to establishing a DOTS session between agents, the owners of
   the networks, domains, services or applications involved are assumed
   to have agreed upon the terms of the relationship involved. Such
   agreements are out of scope for this document, but must be in place
   for a functional DOTS architecture.

   It is assumed that as part of any DOTS service agreement, the DOTS
   client is provided with all data and metadata required to establish
   communication with the DOTS server. Such data and metadata would
   include any cryptographic information necessary to meet the message
   confidentiality, integrity and authenticity requirement in
   [I-D.ietf-dots-requirements], and might also include the pool of DOTS
   server addresses and ports the DOTS client should use for signal and
   data channel messaging.

3.1.2. Establishing the DOTS Session

   With the required business agreements in place, the DOTS client
   initiates a signal session by contacting its DOTS server(s) over the
   signal channel and (possibly) the data channel. To allow for DOTS
   service flexibility, neither the order of contact nor the time
   interval between channel creations is specified. A DOTS client MAY
   establish signal channel first, and then data channel, or vice versa.

The methods by which a DOTS client receives the address and
associated service details of the DOTS server are not prescribed by
this document. For example, a DOTS client may be directly configured
to use a specific DOTS server IP address and port, and directly
provided with any data necessary to satisfy the Peer Mutual
Authentication requirement in [I-D.ietf-dots-requirements], such as
symmetric or asymmetric keys, usernames and passwords, etc. All
configuration and authentication information in this scenario is
provided out-of-band by the domain operating the DOTS server.

At the other extreme, the architecture in this document allows for a
form of DOTS client auto-provisioning. For example, the domain
operating the DOTS server or servers might provide the client domain
only with symmetric or asymmetric keys to authenticate the
provisioned DOTS clients. Only the keys would then be directly
configured on DOTS clients, but the remaining configuration required
to provision the DOTS clients could be learned through mechanisms
similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

The DOTS client SHOULD successfully authenticate and exchange
messages with the DOTS server over both signal and (if used) data
channel as soon as possible to confirm that both channels are
operational.

As described in [I-D.ietf-dots-requirements], the DOTS client can
configure preferred values for acceptable signal loss, mitigation
lifetime, and heartbeat intervals when establishing the DOTS session.
A DOTS session is not active until DOTS agents have agreed on the
values for these DOTS session parameters, a process defined by the
protocol.

Once the DOTS client begins receiving DOTS server signals, the DOTS
session is active. At any time during the DOTS session, the DOTS
client may use the data channel to manage aliases, manage black- and
white-listed prefixes or addresses, leverage vendor-specific
extensions, and so on. Note that unlike the signal channel, there is
no requirement that the data channel remains operational in attack
conditions (See Data Channel Requirements,
[I-D.ietf-dots-requirements]).

3.1.3. Maintaining the DOTS Session

DOTS clients and servers periodically send heartbeats to each other
over the signal channel, per Operational Requirements discussed in
[I-D.ietf-dots-requirements]. DOTS agent operators SHOULD configure
the heartbeat interval such that the frequency does not lead to
accidental denials of service due to the overwhelming number of
heartbeats a DOTS agent must field.

Either DOTS agent may consider a DOTS session terminated in the
extended absence of a heartbeat from its peer agent. The period of
that absence will be established in the protocol definition.

3.2. Modes of Signaling

This section examines the modes of signaling between agents in a DOTS
architecture.

3.2.1. Direct Signaling

A DOTS session may take the form of direct signaling between the DOTS
clients and servers, as shown in Figure 11.

```
         +-------------+                        +-------------+
         | DOTS client |<------signal session------>| DOTS server |
         +-------------+                        +-------------+
```

                       Figure 11: Direct Signaling

In a direct DOTS session, the DOTS client and server are
communicating directly. Direct signaling may exist inter- or intra-
domain. The DOTS session is abstracted from the underlying networks
or network elements the signals traverse: in direct signaling, the
DOTS client and server are logically adjacent.

3.2.2. Redirected Signaling

In certain circumstances, a DOTS server may want to redirect a DOTS
client to an alternative DOTS server for a DOTS session. Such
circumstances include but are not limited to:

o Maximum number of DOTS sessions with clients has been reached;

o Mitigation capacity exhaustion in the mitigator with which the
   specific DOTS server is communicating;

o Mitigator outage or other downtime, such as scheduled maintenance;

o Scheduled DOTS server maintenance;

o Scheduled modifications to the network path between DOTS server
   and DOTS client.

A basic redirected DOTS session resembles the following, as shown in
Figure 12.

```
         +------------+                       +--------------+
         |            |<-(1)--- DOTS session 1 --->|         |
         |            |            |              |          |
         |            |<=(2)== redirect to B ======|         |
         | DOTS client|            |              |DOTS server A|
         |            |X-(4)--- DOTS session 1 ---X|         |
         |            |            |              |          |
         |            |            |              |          |
         +------------+                       +--------------+
               ^
               |
            (3) DOTS session 2
               |
               v
         +--------------+
         | DOTS server B |
         +--------------+
```
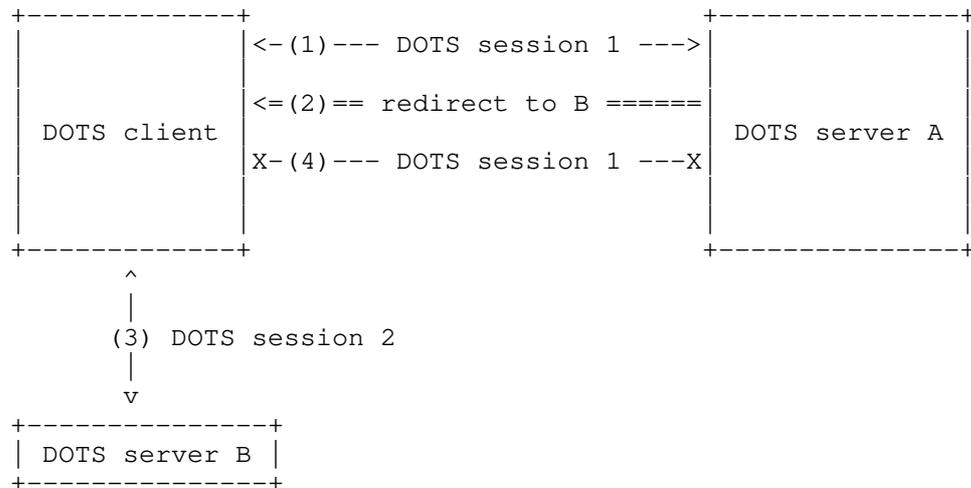
                     Figure 12: Redirected Signaling

   1. Previously established DOTS session 1 exists between a DOTS
       client and DOTS server A.

   2. DOTS server A sends a server signal redirecting the client to
       DOTS server B.

   3. If the DOTS client does not already have a separate DOTS session
       with the redirection target, the DOTS client initiates and
       establishes DOTS session 2 with DOTS server B.

   4. Having redirected the DOTS client, DOTS server A ceases sending
       server signals. The DOTS client likewise stops sending client
       signals to DOTS server A. DOTS session 1 is terminated.

3.2.3. Recursive Signaling

   DOTS is centered around improving the speed and efficiency of
   coordinated response to DDoS attacks. One scenario not yet discussed
   involves coordination among federated domains operating DOTS servers
   and mitigators.

   In the course of normal DOTS operations, a DOTS client communicates
   the need for mitigation to a DOTS server, and that server initiates
   mitigation on a mitigator with which the server has an established
   service relationship. The operator of the mitigator may in turn
   monitor mitigation performance and capacity, as the attack being
   mitigated may grow in severity beyond the mitigating domain's
   capabilities.

The operator of the mitigator has limited options in the event a DOTS client-requested mitigation is being overwhelmed by the severity of the attack. Out-of-scope business or service level agreements may permit the mitigating domain to drop the mitigation and let attack traffic flow unchecked to the target, but this only encourages attack escalation. In the case where the mitigating domain is the upstream service provider for the attack target, this may mean the mitigating domain and its other services and users continue to suffer the incidental effects of the attack.

A recursive signaling model as shown in Figure 13 offers an alternative. In a variation of the use case "End-customer with a single upstream transit provider offering DDoS mitigation services" described in [I-D.ietf-dots-use-cases], a domain operating a DOTS server and mitigator also operates a DOTS client. This DOTS client has an established DOTS session with a DOTS server belonging to a separate administrative domain.

With these preconditions in place, the operator of the mitigator being overwhelmed or otherwise performing inadequately may request mitigation for the attack target from this separate DOTS-aware domain. Such a request recurses the originating mitigation request to the secondary DOTS server, in the hope of building a cumulative mitigation against the attack.

```
                      example.net domain
                . . . . . . . . . . . . . . . .
                .         Gn                   .
      +----+    1 .   +----+    +-----------+  .
      | Cc |<--------->| Sn |~~~~~~~| Mitigator |  .
      +----+    .   +====+    |    Mn     |  .
                .   | Cn |    +-----------+  .
      example.com   .   +----+                   .
         client     .     ^                      .
                . . .|. . . . . . . . . . . . . .
                     |
                  1  |
                     |
                . . .|. . . . . . . . . . . . . .
                .    v                           .
                . +----+    +-----------+        .
                . | So |~~~~~~~| Mitigator |     .
                . +----+    |    Mo     |        .
                .           +-----------+        .
                .                                .
                .                                .
                . . . . . . . . . . . . . . . . .
                      example.org domain
```

Figure 13: Recursive Signaling

In Figure 13, client Cc signals a request for mitigation across
inter-domain DOTS session 1 to the DOTS server Sn belonging to the
example.net domain. DOTS server Sn enables mitigation on mitigator
Mn. DOTS server Sn is half of DOTS gateway Gn, being deployed
logically back-to-back with DOTS client Cn, which has pre-existing
inter-domain DOTS session 2 with the DOTS server So belonging to the
example.org domain. At any point, DOTS server Sn MAY recurse an on-
going mitigation request through DOTS client Cn to DOTS server So, in
the expectation that mitigator Mo will be activated to aid in the
defense of the attack target.

Recursive signaling is opaque to the DOTS client. To maximize
mitigation visibility to the DOTS client, however, the recursing
domain SHOULD provide recursed mitigation feedback in signals
reporting on mitigation status to the DOTS client. For example, the
recursing domain's mitigator should incorporate into mitigation
status messages available metrics such as dropped packet or byte
counts from the recursed mitigation.

DOTS clients involved in recursive signaling must be able to withdraw
requests for mitigation without warning or justification, per
[I-D.ietf-dots-requirements].

Operators recursing mitigation requests MAY maintain the recursed
mitigation for a brief, protocol-defined period in the event the DOTS
client originating the mitigation withdraws its request for help, as
per the discussion of managing mitigation toggling in the operational
requirements ([I-D.ietf-dots-requirements]).

Deployment of recursive signaling may result in traffic redirection,
examination and mitigation extending beyond the initial bilateral
relationship between DOTS client and DOTS server. As such, client
control over the network path of mitigated traffic may be reduced.
DOTS client operators should be aware of any privacy concerns, and
work with DOTS server operators employing recursive signaling to
ensure shared sensitive material is suitably protected.

3.2.4. Anycast Signaling

The DOTS architecture does not assume the availability of anycast
within a DOTS deployment, but neither does the architecture exclude
it. Domains operating DOTS servers MAY deploy DOTS servers with an
anycast Service Address as described in BCP 126 [RFC4786]. In such a
deployment, DOTS clients connecting to the DOTS Service Address may
be communicating with distinct DOTS servers, depending on the network
configuration at the time the DOTS clients connect. Among other
benefits, anycast signaling potentially offers the following:

o Simplified DOTS client configuration, including service discovery
   through the methods described in [RFC7094]. In this scenario, the
   "instance discovery" message would be a DOTS client initiating a
   DOTS session to the DOTS server anycast Service Address, to which
   the DOTS server would reply with a redirection to the DOTS server
   unicast address the client should use for DOTS.

o Region- or customer-specific deployments, in which the DOTS
   Service Addresses route to distinct DOTS servers depending on the
   client region or the customer network in which a DOTS client
   resides.

o Operational resiliency, spreading DOTS signaling traffic across
   the DOTS server domain's networks, and thereby also reducing the
   potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

As long as network configuration remains stable, anycast DOTS
signaling is to the individual DOTS client indistinct from direct
signaling. However, the operational challenges inherent in anycast
signaling are anything but negligible, and DOTS server operators must
carefully weigh the risks against the benefits before deploying.

While the DOTS signal channel primarily operates over UDP per
[I-D.ietf-dots-requirements], the signal channel also requires mutual
authentication between DOTS agents, with associated security state on
both ends.

Network instability is of particular concern with anycast signaling,
as DOTS signal channels are expected to be long-lived, and
potentially operating under congested network conditions caused by a
volumetric DDoS attack.

For example, a network configuration altering the route to the DOTS
server during active anycast signaling may cause the DOTS client to
send messages to a DOTS server other than the one with which it
initially established a signaling session. That second DOTS server
may not have the security state of the existing session, forcing the
DOTS client to initialize a new DOTS session. This challenge might
in part be mitigated by use of pre-shared keys and session resumption
[RFC5246][RFC6347], but keying material must be available to all DOTS
servers sharing the anycast Service Address in that case.

While the DOTS client will try to establish a new DOTS session with
the DOTS server now acting as the anycast DOTS Service Address, the
link between DOTS client and server may be congested with attack
traffic, making signal session establishment difficult. In such a
scenario, anycast Service Address instability becomes a sort of
signal session flapping, with obvious negative consequences for the
DOTS deployment.

Anycast signaling deployments similarly must also take into account
active mitigations. Active mitigations initiated through a DOTS
session may involve diverting traffic to a scrubbing center. If the
DOTS session flaps due to anycast changes as described above,
mitigation may also flap as the DOTS servers sharing the anycast DOTS
service address toggles mitigation on detecting DOTS session loss,
depending on whether the client has configured mitigation on loss of
signal.

3.3. Triggering Requests for Mitigation

[I-D.ietf-dots-requirements] places no limitation on the
circumstances in which a DOTS client operator may request mitigation,
nor does it demand justification for any mitigation request, thereby
reserving operational control over DDoS defense for the domain
requesting mitigation. This architecture likewise does not prescribe
the network conditions and mechanisms triggering a mitigation request
from a DOTS client.

However, considering selected possible mitigation triggers from an
architectural perspective offers a model for alternative or
unanticipated triggers for DOTS deployments. In all cases, what
network conditions merit a mitigation request are at the discretion
of the DOTS client operator.

The mitigation request itself is defined by DOTS, however the
interfaces required to trigger the mitigation request in the
following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

A DOTS client operator may manually prepare a request for mitigation,
including scope and duration, and manually instruct the DOTS client
to send the mitigation request to the DOTS server. In context, a
manual request is a request directly issued by the operator without
automated decision-making performed by a device interacting with the
DOTS client. Modes of manual mitigation requests include an operator
entering a command into a text interface, or directly interacting
with a graphical interface to send the request.

An operator might do this, for example, in response to notice of an
attack delivered by attack detection equipment or software, and the
alerting detector lacks interfaces or is not configured to use
available interfaces to translate the alert to a mitigation request
automatically.

In a variation of the above scenario, the operator may have
preconfigured on the DOTS client mitigation requests for various
resources in the operator's domain. When notified of an attack, the
DOTS client operator manually instructs the DOTS client to send the
relevant preconfigured mitigation request for the resources under
attack.

A further variant involves recursive signaling, as described in
Section 3.2.3. The DOTS client in this case is the second half of a
DOTS gateway (back-to-back DOTS server and client). As in the
previous scenario, the scope and duration of the mitigation request
are pre-existing, but in this case are derived from the mitigation
request received from a downstream DOTS client by the DOTS server.
Assuming the preconditions required by Section 3.2.3 are in place,
the DOTS gateway operator may at any time manually request mitigation
from an upstream DOTS server, sending a mitigation request derived
from the downstream DOTS client's request.

The motivations for a DOTS client operator to request mitigation
manually are not prescribed by this architecture, but are expected to
include some of the following:

   o Notice of an attack delivered via e-mail or alternative messaging

   o Notice of an attack delivered via phone call

   o Notice of an attack delivered through the interface(s) of
      networking monitoring software deployed in the operator's domain

   o Manual monitoring of network behavior through network monitoring
      software

3.3.2. Automated Conditional Mitigation Request

   Unlike manual mitigation requests, which depend entirely on the DOTS
   client operator's capacity to react with speed and accuracy to every
   detected or detectable attack, mitigation requests triggered by
   detected attack conditions reduce the operational burden on the DOTS
   client operator, and minimize the latency between attack detection
   and the start of mitigation.

   Mitigation requests are triggered in this scenario by operator-
   specified network conditions. Attack detection is deployment-
   specific, and not constrained by this architecture. Similarly the
   specifics of a condition are left to the discretion of the operator,
   though common conditions meriting mitigation include the following:

   o Detected attack exceeding a rate in packets per second (pps).

   o Detected attack exceeding a rate in bytes per second (bps).

   o Detected resource exhaustion in an attack target.

   o Detected resource exhaustion in the local domain's mitigator.

   o Number of open connections to an attack target.

   o Number of attack sources in a given attack.

   o Number of active attacks against targets in the operator's domain.

   o Conditional detection developed through arbitrary statistical
      analysis or deep learning techniques.

   o Any combination of the above.

   When automated conditional mitigation requests are enabled,
   violations of any of the above conditions, or any additional
   operator-defined conditions, will trigger a mitigation request from
   the DOTS client to the DOTS server. The interfaces between the

application detecting the condition violation and the DOTS client are
implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

To maintain a DOTS session, the DOTS client and the DOTS server
exchange regular but infrequent messages across the signal channel.
In the absence of an attack, the probability of message loss in the
signaling channel should be extremely low. Under attack conditions,
however, some signal loss may be anticipated as attack traffic
congests the link, depending on the attack type.

While [I-D.ietf-dots-requirements] specifies the DOTS protocol be
robust when signaling under attack conditions, there are nevertheless
scenarios in which the DOTS signal is lost in spite of protocol best
efforts. To handle such scenarios, a DOTS operator may configure the
DOTS session to trigger mitigation when the DOTS server ceases
receiving DOTS client signals (or vice versa) beyond the miss count
or period permitted by the protocol.

The impact of mitigating due to loss of signal in either direction
must be considered carefully before enabling it. Signal loss is not
caused by links congested with attack traffic alone, and as such
mitigation requests triggered by signal channel degradation in either
direction may incur unnecessary costs, in network performance and
operational expense alike.

4. Security Considerations

This section describes identified security considerations for the
DOTS architecture.

DOTS is at risk from three primary attack vectors: agent
impersonation, traffic injection and signal blocking. These vectors
may be exploited individually or in concert by an attacker to
confuse, disable, take information from, or otherwise inhibit DOTS
agents.

Any attacker with the ability to impersonate a legitimate DOTS client
or server or, indeed, inject false messages into the stream may
potentially trigger/withdraw traffic redirection, trigger/cancel
mitigation activities or subvert black/whitelists. From an
architectural standpoint, operators SHOULD ensure best current
practices for secure communication are observed for data and signal
channel confidentiality, integrity and authenticity. Care must be
taken to ensure transmission is protected by appropriately secure
means, reducing attack surface by exposing only the minimal required

services or interfaces. Similarly, received data at rest SHOULD be
stored with a satisfactory degree of security.

As many mitigation systems employ diversion to scrub attack traffic,
operators of DOTS agents SHOULD ensure DOTS sessions are resistant to
Man-in-the-Middle (MitM) attacks. An attacker with control of a DOTS
client may negatively influence network traffic by requesting and
withdrawing requests for mitigation for particular prefixes, leading
to route or DNS flapping.

Any attack targeting the availability of DOTS servers may disrupt the
ability of the system to receive and process DOTS signals resulting
in failure to fulfill a mitigation request. DOTS agents SHOULD be
given adequate protections, again in accordance with best current
practices for network and host security.

5. Contributors

   Mohamed Boucadair
      Orange

      mohamed.boucadair@orange.com

6. Acknowledgments

   Thanks to Matt Richardson and Med Boucadair for their comments and
   suggestions.

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>.

7.2. Informative References

   [I-D.ietf-dots-requirements]
              Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed
              Denial of Service (DDoS) Open Threat Signaling
              Requirements", draft-ietf-dots-requirements-06 (work in
              progress), July 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-07 (work
              in progress), July 2017.

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

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

   [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>.

   [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
              specifying the location of services (DNS SRV)", RFC 2782,
              DOI 10.17487/RFC2782, February 2000,
              <https://www.rfc-editor.org/info/rfc2782>.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A
              Border Gateway Protocol 4 (BGP-4)", RFC 4271,
              DOI 10.17487/RFC4271, January 2006,
              <https://www.rfc-editor.org/info/rfc4271>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [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>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC6347]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer
               Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
               January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC6763]   Cheshire, S. and M. Krochmal, "DNS-Based Service
               Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
               <https://www.rfc-editor.org/info/rfc6763>.

   [RFC7092]   Kaplan, H. and V. Pascual, "A Taxonomy of Session
               Initiation Protocol (SIP) Back-to-Back User Agents",
               RFC 7092, DOI 10.17487/RFC7092, December 2013,
               <https://www.rfc-editor.org/info/rfc7092>.

   [RFC7094]   McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
               "Architectural Considerations of IP Anycast", RFC 7094,
               DOI 10.17487/RFC7094, January 2014,
               <https://www.rfc-editor.org/info/rfc7094>.

Authors' Addresses

   Andrew Mortensen
   Arbor Networks
   2727 S. State St
   Ann Arbor, MI 48104
   United States

   EMail: amortensen@arbor.net

   Flemming Andreasen
   Cisco
   United States

   EMail: fandreas@cisco.com

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

   EMail: tireddy@cisco.com

Christopher Gray
Comcast
United States

EMail: Christopher_Gray3@cable.comcast.com

Rich Compton
Charter

EMail: Rich.Compton@charter.com

Nik Teague
Verisign
United States

EMail: nteague@verisign.com

        Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
                      draft-ietf-dots-architecture-14

   Abstract

      This document describes an architecture for establishing and
      maintaining Distributed Denial of Service (DDoS) Open Threat
      Signaling (DOTS) within and between domains. The document does not
      specify protocols or protocol extensions, instead focusing on
      defining architectural relationships, components and concepts used in
      a DOTS deployment.

   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 30, 2019.

   Copyright Notice

Table of Contents

1. Context and Motivation

   Signaling the need for help defending against an active distributed
   denial of service (DDoS) attack requires a common understanding of
   mechanisms and roles among the parties coordinating defensive
   response. The signaling layer and supplementary messaging is the
   focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of
   coordinating defensive measures among willing peers to mitigate
   attacks quickly and efficiently, enabling hybrid attack responses
   coordinated locally at or near the target of an active attack, or
   anywhere in-path between attack sources and target. Sample DOTS use
   cases are elaborated in [I-D.ietf-dots-use-cases].

   This document describes an architecture used in establishing,
   maintaining or terminating a DOTS relationship within a domain or
   between domains.

1.1. Terminology

1.1.1. Key Words

   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 BCP14 [RFC2119]
   [RFC8174], when, and only when, they appear in all capitals.

1.1.2. Definition of Terms

   This document uses the terms defined in [RFC8612].

1.2. Scope

   In this architecture, DOTS clients and servers communicate using DOTS
   signaling. As a result of signals from a DOTS client, the DOTS
   server may modify the forwarding path of traffic destined for the
   attack target(s), for example by diverting traffic to a mitigator or
   pool of mitigators, where policy may be applied to distinguish and
   discard attack traffic. Any such policy is deployment-specific.

   The DOTS architecture presented here is applicable across network
   administrative domains - for example, between an enterprise domain
   and the domain of a third-party attack mitigation service - as well
   as to a single administrative domain. DOTS is generally assumed to
   be most effective when aiding coordination of attack response between
   two or more participating networks, but single domain scenarios are

valuable in their own right, as when aggregating intra-domain DOTS
client signals for inter-domain coordinated attack response.

This document does not address any administrative or business
agreements that may be established between involved DOTS parties.
Those considerations are out of scope. Regardless, this document
assumes necessary authentication and authorization mechanisms are put
in place so that only authorized clients can invoke the DOTS service.

A detailed set of DOTS requirements are discussed in [RFC8612], and
the DOTS architecture is designed to follow those requirements. Only
new behavioral requirements are described in this document.

1.3. Assumptions

This document makes the following assumptions:

o All domains in which DOTS is deployed are assumed to offer the
   required connectivity between DOTS agents and any intermediary
   network elements, but the architecture imposes no additional
   limitations on the form of connectivity.

o Congestion and resource exhaustion are intended outcomes of a DDoS
   attack [RFC4732]. Some operators may utilize non-impacted paths
   or networks for DOTS, but in general conditions should be assumed
   to be hostile and DOTS must be able to function in all
   circumstances, including when the signaling path is significantly
   impaired.

o There is no universal DDoS attack scale threshold triggering a
   coordinated response across administrative domains. A network
   domain administrator, or service or application owner may
   arbitrarily set attack scale threshold triggers, or manually send
   requests for mitigation.

o Mitigation requests may be sent to one or more upstream DOTS
   servers based on criteria determined by DOTS client administrators
   and the underlying network configuration. The number of DOTS
   servers with which a given DOTS client has established
   communications is determined by local policy and is deployment-
   specific. For example, a DOTS client of a multi-homed network may
   support built-in policies to establish DOTS relationships with
   DOTS servers located upstream of each interconnection link.

o The mitigation capacity and/or capability of domains receiving
   requests for coordinated attack response is opaque to the domains
   sending the request. The domain receiving the DOTS client signal
   may or may not have sufficient capacity or capability to filter

any or all DDoS attack traffic directed at a target. In either
case, the upstream DOTS server may redirect a request to another
DOTS server. Redirection may be local to the redirecting DOTS
server's domain, or may involve a third-party domain.

o DOTS client and server signals, as well as messages sent through
   the data channel, are sent across any transit networks with the
   same probability of delivery as any other traffic between the DOTS
   client domain and the DOTS server domain. Any encapsulation
   required for successful delivery is left untouched by transit
   network elements. DOTS server and DOTS client cannot assume any
   preferential treatment of DOTS signals. Such preferential
   treatment may be available in some deployments (e.g., intra-domain
   scenarios), and the DOTS architecture does not preclude its use
   when available. However, DOTS itself does not address how that
   may be done.

o The architecture allows for, but does not assume, the presence of
   Quality of Service (QoS) policy agreements between DOTS-enabled
   peer networks or local QoS prioritization aimed at ensuring
   delivery of DOTS messages between DOTS agents. QoS is an
   operational consideration only, not a functional part of the DOTS
   architecture.

o The signal and data channels are loosely coupled, and may not
   terminate on the same DOTS server.

2. DOTS Architecture

   The basic high-level DOTS architecture is illustrated in Figure 1:

```
      +-----------+                 +-------------+
      | Mitigator | ~~~~~~~~~~      | DOTS Server |
      +-----------+                 +-------------+
                                           |
                                           |
                                           |
                                           |
      +---------------+                +-------------+
      | Attack Target | ~~~~~~        | DOTS Client |
      +---------------+                +-------------+
```

                    Figure 1: Basic DOTS Architecture

   A simple example instantiation of the DOTS architecture could be an
   enterprise as the attack target for a volumetric DDoS attack, and an
   upstream DDoS mitigation service as the mitigator. The enterprise
   (attack target) is connected to the Internet via a link that is
   getting saturated, and the enterprise suspects it is under DDoS

attack. The enterprise has a DOTS client, which obtains information
about the DDoS attack, and signals the DOTS server for help in
mitigating the attack. The DOTS server in turn invokes one or more
mitigators, which are tasked with mitigating the actual DDoS attack,
and hence aim to suppress the attack traffic while allowing valid
traffic to reach the attack target.

The scope of the DOTS specifications is the interfaces between the
DOTS client and DOTS server. The interfaces to the attack target and
the mitigator are out of scope of DOTS. Similarly, the operation of
both the attack target and the mitigator is out of scope of DOTS.
Thus, DOTS neither specifies how an attack target decides it is under
DDoS attack, nor does DOTS specify how a mitigator may actually
mitigate such an attack. A DOTS client's request for mitigation is
advisory in nature, and may not lead to any mitigation at all,
depending on the DOTS server domain's capacity and willingness to
mitigate on behalf of the DOTS client's domain.

The DOTS client may be provided with a list of DOTS servers, each
associated with one or more IP addresses. These addresses may or may
not be of the same address family. The DOTS client establishes one
or more sessions by connecting to the provided DOTS server addresses.

As illustrated in Figure 2, there are two interfaces between a DOTS
server and a DOTS client; a signal channel and (optionally) a data
channel.

```
+---------------+                               +---------------+
|               | <------- Signal Channel ------> |               |
|  DOTS Client  |                               |  DOTS Server  |
|               | <=======  Data Channel  ======> |               |
+---------------+                               +---------------+
```

                      Figure 2: DOTS Interfaces

The primary purpose of the signal channel is for a DOTS client to ask
a DOTS server for help in mitigating an attack, and for the DOTS
server to inform the DOTS client about the status of such mitigation.
The DOTS client does this by sending a client signal, which contains
information about the attack target(s). The client signal may also
include telemetry information about the attack, if the DOTS client
has such information available. The DOTS server in turn sends a
server signal to inform the DOTS client of whether it will honor the
mitigation request. Assuming it will, the DOTS server initiates
attack mitigation, and periodically informs the DOTS client about the
status of the mitigation. Similarly, the DOTS client periodically
informs the DOTS server about the client's status, which at a minimum
provides client (attack target) health information, but it should

also include efficacy information about the attack mitigation as it
is now seen by the client. At some point, the DOTS client may decide
to terminate the server-side attack mitigation, which it indicates to
the DOTS server over the signal channel. A mitigation may also be
terminated if a DOTS client-specified mitigation lifetime is
exceeded. Note that the signal channel may need to operate over a
link that is experiencing a DDoS attack and hence is subject to
severe packet loss and high latency.

While DOTS is able to request mitigation with just the signal
channel, the addition of the DOTS data channel provides for
additional and more efficient capabilities. The primary purpose of
the data channel is to support DOTS related configuration and policy
information exchange between the DOTS client and the DOTS server.
Examples of such information include, but are not limited to:

o Creating identifiers, such as names or aliases, for resources for
   which mitigation may be requested. Such identifiers may then be
   used in subsequent signal channel exchanges to refer more
   efficiently to the resources under attack, as seen in Figure 3,
   using JSON to serialize the data:

```
{
    "https1": [
        "192.0.2.1:443",
        "198.51.100.2:443",
    ],
    "proxies": [
        "203.0.113.3:3128",
        "[2001:db8:ac10::1]:3128"
    ],
    "api_urls": "https://apiserver.example.com/api/v1",
}
```

            Figure 3: Protected resource identifiers

o Drop-list management, which enables a DOTS client to inform the
   DOTS server about sources to suppress.

o Accept-list management, which enables a DOTS client to inform the
   DOTS server about sources from which traffic is always accepted.

o Filter management, which enables a DOTS client to install or
   remove traffic filters dropping or rate-limiting unwanted traffic.

o DOTS client provisioning.

Note that while it is possible to exchange the above information
before, during or after a DDoS attack, DOTS requires reliable
delivery of this information and does not provide any special means
for ensuring timely delivery of it during an attack. In practice,
this means that DOTS deployments should not rely on such information
being exchanged during a DDoS attack.

2.1. DOTS Operations

DOTS does not prescribe any specific deployment models, however DOTS
is designed with some specific requirements around the different DOTS
agents and their relationships.

First of all, a DOTS agent belongs to a domain that has an identity
which can be authenticated and authorized. DOTS agents communicate
with each other over a mutually authenticated signal channel and
(optionally) data channel. However, before they can do so, a service
relationship needs to be established between them. The details and
means by which this is done is outside the scope of DOTS, however an
example would be for an enterprise A (DOTS client) to sign up for
DDoS service from provider B (DOTS server). This would establish a
(service) relationship between the two that enables enterprise A's
DOTS client to establish a signal channel with provider B's DOTS
server. A and B will authenticate each other, and B can verify that
A is authorized for its service.

From an operational and design point of view, DOTS assumes that the
above relationship is established prior to a request for DDoS attack
mitigation. In particular, it is assumed that bi-directional
communication is possible at this time between the DOTS client and
DOTS server. Furthermore, it is assumed that additional service
provisioning, configuration and information exchange can be performed
by use of the data channel, if operationally required. It is not
until this point that the mitigation service is available for use.

Once the mutually authenticated signal channel has been established,
it will remain active. This is done to increase the likelihood that
the DOTS client can signal the DOTS server for help when the attack
target is being flooded, and similarly raise the probability that
DOTS server signals reach the client regardless of inbound link
congestion. This does not necessarily imply that the attack target
and the DOTS client have to be co-located in the same administrative
domain, but it is expected to be a common scenario.

DDoS mitigation with the help of an upstream mitigator may involve
some form of traffic redirection whereby traffic destined for the
attack target is steered towards the mitigator. Common mechanisms to
achieve this redirection depend on BGP [RFC4271] and DNS [RFC1035].

The mitigator in turn inspects and scrubs the traffic, and forwards
the resulting (hopefully non-attack) traffic to the attack target.
Thus, when a DOTS server receives an attack mitigation request from a
DOTS client, it can be viewed as a way of causing traffic redirection
for the attack target indicated.

DOTS relies on mutual authentication and the pre-established service
relationship between the DOTS client's domain and the DOTS server's
domain to provide basic authorization. The DOTS server should
enforce additional authorization mechanisms to restrict the
mitigation scope a DOTS client can request, but such authorization
mechanisms are deployment-specific.

Although co-location of DOTS server and mitigator within the same
domain is expected to be a common deployment model, it is assumed
that operators may require alternative models. Nothing in this
document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help
coordinating attack response originate. The requests may be in
response to an active, ongoing attack against a target in the DOTS
client's domain, but no active attack is required for a DOTS client
to request help. Operators may wish to have upstream mitigators in
the network path for an indefinite period, and are restricted only by
business relationships when it comes to duration and scope of
requested mitigation.

The DOTS client requests attack response coordination from a DOTS
server over the signal channel, including in the request the DOTS
client's desired mitigation scoping, as described in [RFC8612] (SIG-
008). The actual mitigation scope and countermeasures used in
response to the attack are up to the DOTS server and mitigator
operators, as the DOTS client may have a narrow perspective on the
ongoing attack. As such, the DOTS client's request for mitigation
should be considered advisory: guarantees of DOTS server availability
or mitigation capacity constitute service level agreements and are
out of scope for this document.

The DOTS client adjusts mitigation scope and provides available
mitigation feedback (e.g., mitigation efficacy) at the direction of
its local administrator. Such direction may involve manual or
automated adjustments in response to updates from the DOTS server.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct session, the DOTS client sends
a heartbeat over the signal channel to maintain its half of the
channel. The DOTS client similarly expects a heartbeat from the DOTS
server, and may consider a session terminated in the extended absence
of a DOTS server heartbeat.

2.2.2. DOTS Server

A DOTS server is a DOTS agent capable of receiving, processing and
possibly acting on requests for help coordinating attack response
from DOTS clients. The DOTS server authenticates and authorizes DOTS
clients as described in Section 3.1, and maintains session state,
tracking requests for mitigation, reporting on the status of active
mitigations, and terminating sessions in the extended absence of a
client heartbeat or when a session times out.

Assuming the preconditions discussed below exist, a DOTS client
maintaining an active session with a DOTS server may reasonably
expect some level of mitigation in response to a request for
coordinated attack response.

For a given DOTS client (administrative) domain, the DOTS server
needs to be able to determine whether a given target resource is in
that domain. For example, this could take the form of associating a
set of IP addresses and/or prefixes per domain. The DOTS server
enforces authorization of DOTS clients' signals for mitigation. The
mechanism of enforcement is not in scope for this document, but is
expected to restrict requested mitigation scope to addresses,
prefixes, and/or services owned by the DOTS client domain, such that
a DOTS client from one domain is not able to influence the network
path to another domain. A DOTS server MUST reject requests for
mitigation of resources not owned by the requesting DOTS client's
administrative domain. A DOTS server MAY also refuse a DOTS client's
mitigation request for arbitrary reasons, within any limits imposed
by business or service level agreements between client and server
domains. If a DOTS server refuses a DOTS client's request for
mitigation, the DOTS server MUST include the refusal reason in the
server signal sent to the client.

A DOTS server is in regular contact with one or more mitigators. If
a DOTS server accepts a DOTS client's request for help, the DOTS
server forwards a translated form of that request to the mitigator(s)
responsible for scrubbing attack traffic. Note that the form of the
translated request passed from the DOTS server to the mitigator is
not in scope: it may be as simple as an alert to mitigator operators,
or highly automated using vendor or open application programming

interfaces supported by the mitigator. The DOTS server MUST report
the actual scope of any mitigation enabled on behalf of a client.

The DOTS server SHOULD retrieve available metrics for any mitigations
activated on behalf of a DOTS client, and SHOULD include them in
server signals sent to the DOTS client originating the request for
mitigation.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct channel, the DOTS server MUST
send a heartbeat over the signal channel to maintain its half of the
channel. The DOTS server similarly expects a heartbeat from the DOTS
client, and MAY consider a session terminated in the extended absence
of a DOTS client heartbeat.

2.2.3. DOTS Gateway

Traditional client/server relationships may be expanded by chaining
DOTS sessions. This chaining is enabled through "logical
concatenation" of a DOTS server and a DOTS client, resulting in an
application analogous to the Session Initiation Protocol (SIP)
[RFC3261] logical entity of a Back-to-Back User Agent (B2BUA)
[RFC7092]. The term DOTS gateway is used here in the descriptions of
selected scenarios involving this application.

A DOTS gateway may be deployed client-side, server-side or both. The
gateway may terminate multiple discrete client connections and may
aggregate these into a single or multiple DOTS sessions.

The DOTS gateway will appear as a server to its downstream agents and
as a client to its upstream agents, a functional concatenation of the
DOTS client and server roles, as depicted in Figure 4:

```
                      +-------------+
                      |     | D |   |
      +----+          |     | O |   |          +----+
      | c1 |----------| s1  | T | c2 |---------| s2 |
      +----+          |     | S |   |          +----+
                      |     | G |   |
                      +-------------+
```

                     Figure 4: DOTS gateway

The DOTS gateway MUST perform full stack DOTS session termination and
reorigination between its client and server side. The details of how
this is achieved are implementation specific. The DOTS protocol does
not include any special features related to DOTS gateways, and hence

   from a DOTS perspective, whenever a DOTS gateway is present, the DOTS
   session simply terminates/originates there.

2.3. DOTS Agent Relationships

   So far, we have only considered a relatively simple scenario of a
   single DOTS client associated with a single DOTS server, however DOTS
   supports more advanced relationships.

   A DOTS server may be associated with one or more DOTS clients, and
   those DOTS clients may belong to different domains. An example
   scenario is a mitigation provider serving multiple attack targets
   (Figure 5).

```
        DOTS clients        DOTS server
        +---+
        | c |-----------
        +---+           \
        c1.example.org    \
                            \
        +---+               \ +---+
        | c |---------------| S |
        +---+               / +---+
        c1.example.com    /   dots1.example.net
                        /
        +---+         /
        | c |-----------
        +---+
        c2.example.com
```

                 Figure 5: DOTS server with multiple clients

   A DOTS client may be associated with one or more DOTS servers, and
   those DOTS servers may belong to different domains. This may be to
   ensure high availability or co-ordinate mitigation with more than one
   directly connected ISP. An example scenario is for an enterprise to
   have DDoS mitigation service from multiple providers, as shown in
   Figure 6.

```
    DOTS client        DOTS servers
                          +---+
               ----------| S |
              /           +---+
             /            dots1.example.net
            /
   +---+/           +---+
   | c |--------------| S |
   +---+\           +---+
        \            dots.example.org
         \
          \          +---+
           ----------| S |
                      +---+
    c.example.com    dots2.example.net
```

                  Figure 6: Multi-Homed DOTS Client

   Deploying a multi-homed client requires extra care and planning, as
   the DOTS servers with which the multi-homed client communicates may
   not be affiliated. Should the multi-homed client simultaneously
   request for mitigation from all servers with which it has established
   signal channels, the client may unintentionally inflict additional
   network disruption on the resources it intends to protect. In one of
   the worst cases, a multi-homed DOTS client could cause a permanent
   routing loop of traffic destined for the client's protected services,
   as the uncoordinated DOTS servers' mitigators all try to divert that
   traffic to their own scrubbing centers.

   The DOTS protocol itself provides no fool-proof method to prevent
   such self-inflicted harms as a result deploying multi-homed DOTS
   clients. If DOTS client implementations nevertheless include support
   for multi-homing, they are expected to be aware of the risks, and
   consequently to include measures aimed at reducing the likelihood of
   negative outcomes. Simple measures might include:

   o Requesting mitigation serially, ensuring only one mitigation
      request for a given address space is active at any given time;

   o Dividing the protected resources among the DOTS servers, such that
      no two mitigators will be attempting to divert and scrub the same
      traffic;

   o Restricting multi-homing to deployments in which all DOTS servers
      are coordinating management of a shared pool of mitigation
      resources.

2.3.1. Gatewayed Signaling

   As discussed in Section 2.2.3, a DOTS gateway is a logical function
   chaining DOTS sessions through concatenation of a DOTS server and
   DOTS client.

   An example scenario, as shown in Figure 7 and Figure 8, is for an
   enterprise to have deployed multiple DOTS capable devices which are
   able to signal intra-domain using TCP [RFC0793] on un-congested links
   to a DOTS gateway which may then transform these to a UDP [RFC0768]
   transport inter-domain where connection oriented transports may
   degrade; this applies to the signal channel only, as the data channel
   requires a connection-oriented transport. The relationship between
   the gateway and its upstream agents is opaque to the initial clients.

```
            +---+
            | c |\
            +---+ \                +---+
                   \-----TCP-----| D |                +---+
            +---+                  | O |                |   |
            | c |--------TCP-----| T |------UDP------| S |
            +---+                  | S |                |   |
                   /-----TCP-----| G |                +---+
            +---+ /                +---+
            | c |/
            +---+
            example.com        example.com          example.net
            DOTS clients       DOTS gateway (DOTSG)  DOTS server
```

              Figure 7: Client-Side Gateway with Aggregation

```
            +---+
            | c |\
            +---+ \                +---+
                   \-----TCP-----| D |------UDP------+---+
            +---+                  | O |                |   |
            | c |--------TCP-----| T |------UDP------| S |
            +---+                  | S |                |   |
                   /-----TCP-----| G |------UDP------+---+
            +---+ /                +---+
            | c |/
            +---+
            example.com        example.com          example.net
            DOTS clients       DOTS gateway (DOTSG)  DOTS server
```

             Figure 8: Client-Side Gateway without Aggregation

This may similarly be deployed in the inverse scenario where the
gateway resides in the server-side domain and may be used to
terminate and/or aggregate multiple clients to single transport as
shown in figures Figure 9 and Figure 10.

```
        +---+
        | c |\
        +---+ \            +---+
               \-----UDP-----| D |            +---+
        +---+                | O |            |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+                | S |            |   |
               /-----TCP-----| G |            +---+
        +---+ /              +---+
        | c |/
        +---+
        example.com       example.net         example.net
        DOTS clients      DOTS gateway (DOTSG) DOTS server
```

             Figure 9: Server-Side Gateway with Aggregation

```
        +---+
        | c |\
        +---+ \            +---+
               \-----UDP-----| D |------TCP------+---+
        +---+                | O |            |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+                | S |            |   |
               /-----UDP-----| G |------TCP------+---+
        +---+ /              +---+
        | c |/
        +---+
        example.com       example.net         example.net
        DOTS clients      DOTS gateway (DOTSG) DOTS server
```

            Figure 10: Server-Side Gateway without Aggregation

   This document anticipates scenarios involving multiple DOTS gateways.
   An example is a DOTS gateway at the network client's side, and
   another one at the server side. The first gateway can be located at
   a CPE to aggregate requests from multiple DOTS clients enabled in an
   enterprise network. The second DOTS gateway is deployed on the
   provider side. This scenario can be seen as a combination of the
   client-side and server-side scenarios.

3. Concepts

3.1. DOTS Sessions

   In order for DOTS to be effective as a vehicle for DDoS mitigation
   requests, one or more DOTS clients must establish ongoing
   communication with one or more DOTS servers. While the preconditions
   for enabling DOTS in or among network domains may also involve
   business relationships, service level agreements, or other formal or
   informal understandings between network operators, such
   considerations are out of scope for this document.

   A DOTS session is established to support bilateral exchange of data
   between an associated DOTS client and a DOTS server. In the DOTS
   architecture, data is exchanged between DOTS agents over signal and
   data channels. As such, a DOTS session can be a DOTS signal channel
   session, a DOTS data channel session, or both.

   A DOTS agent can maintain one or more DOTS sessions.

   A DOTS signal channel session is associated with a single transport
   connection (TCP or UDP session) and an ephemeral security association
   (a TLS or DTLS session). Similarly, a DOTS data channel session is
   associated with a single TCP connection and an ephemeral TLS security
   association.

   Mitigation requests created using DOTS signal channel are not bound
   to the DOTS signal channel session. Instead, mitigation requests are
   associated with a DOTS client and can be managed using different DOTS
   signal channel sessions.

3.1.1. Preconditions

   Prior to establishing a DOTS session between agents, the owners of
   the networks, domains, services or applications involved are assumed
   to have agreed upon the terms of the relationship involved. Such
   agreements are out of scope for this document, but must be in place
   for a functional DOTS architecture.

   It is assumed that as part of any DOTS service agreement, the DOTS
   client is provided with all data and metadata required to establish
   communication with the DOTS server. Such data and metadata would
   include any cryptographic information necessary to meet the message
   confidentiality, integrity and authenticity requirement (SEC-002) in
   [RFC8612], and might also include the pool of DOTS server addresses
   and ports the DOTS client should use for signal and data channel
   messaging.

3.1.2. Establishing the DOTS Session

   With the required business agreements in place, the DOTS client
   initiates a DOTS session by contacting its DOTS server(s) over the
   signal channel and (possibly) the data channel. To allow for DOTS
   service flexibility, neither the order of contact nor the time
   interval between channel creations is specified. A DOTS client MAY
   establish signal channel first, and then data channel, or vice versa.

   The methods by which a DOTS client receives the address and
   associated service details of the DOTS server are not prescribed by
   this document. For example, a DOTS client may be directly configured
   to use a specific DOTS server IP address and port, and directly
   provided with any data necessary to satisfy the Peer Mutual
   Authentication requirement (SEC-001) in [RFC8612], such as symmetric
   or asymmetric keys, usernames and passwords, etc. All configuration
   and authentication information in this scenario is provided out-of-
   band by the domain operating the DOTS server.

   At the other extreme, the architecture in this document allows for a
   form of DOTS client auto-provisioning. For example, the domain
   operating the DOTS server or servers might provide the client domain
   only with symmetric or asymmetric keys to authenticate the
   provisioned DOTS clients. Only the keys would then be directly
   configured on DOTS clients, but the remaining configuration required
   to provision the DOTS clients could be learned through mechanisms
   similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

   The DOTS client SHOULD successfully authenticate and exchange
   messages with the DOTS server over both signal and (if used) data
   channel as soon as possible to confirm that both channels are
   operational.

   As described in [RFC8612] (DM-008), the DOTS client can configure
   preferred values for acceptable signal loss, mitigation lifetime, and
   heartbeat intervals when establishing the DOTS signal channel
   session. A DOTS signal channel session is not active until DOTS
   agents have agreed on the values for these DOTS session parameters, a
   process defined by the protocol.

   Once the DOTS client begins receiving DOTS server signals, the DOTS
   session is active. At any time during the DOTS session, the DOTS
   client may use the data channel to manage aliases, manage drop- and
   accept-listed prefixes or addresses, leverage vendor-specific
   extensions, and so on. Note that unlike the signal channel, there is
   no requirement that the data channel remains operational in attack
   conditions (See Data Channel Requirements, Section 2.3 of [RFC8612]).

3.1.3. Maintaining the DOTS Session

   DOTS clients and servers periodically send heartbeats to each other
   over the signal channel, discussed in [RFC8612] (SIG-004). DOTS
   agent operators SHOULD configure the heartbeat interval such that the
   frequency does not lead to accidental denials of service due to the
   overwhelming number of heartbeats a DOTS agent must field.

   Either DOTS agent may consider a DOTS signal channel session
   terminated in the extended absence of a heartbeat from its peer
   agent. The period of that absence will be established in the
   protocol definition.

3.2. Modes of Signaling

   This section examines the modes of signaling between agents in a DOTS
   architecture.

3.2.1. Direct Signaling

   A DOTS session may take the form of direct signaling between the DOTS
   clients and servers, as shown in Figure 11.

```
        +-------------+                          +-------------+
        | DOTS client |<------signal session------>| DOTS server |
        +-------------+                          +-------------+
```

                     Figure 11: Direct Signaling

   In a direct DOTS session, the DOTS client and server are
   communicating directly. Direct signaling may exist inter- or intra-
   domain. The DOTS session is abstracted from the underlying networks
   or network elements the signals traverse: in direct signaling, the
   DOTS client and server are logically adjacent.

3.2.2. Redirected Signaling

   In certain circumstances, a DOTS server may want to redirect a DOTS
   client to an alternative DOTS server for a DOTS signal channel
   session. Such circumstances include but are not limited to:

   o Maximum number of DOTS signal channel sessions with clients has
      been reached;

   o Mitigation capacity exhaustion in the mitigator with which the
      specific DOTS server is communicating;

   o Mitigator outage or other downtime, such as scheduled maintenance;

o Scheduled DOTS server maintenance;

o Scheduled modifications to the network path between DOTS server
   and DOTS client.

A basic redirected DOTS signal channel session resembles the
following, as shown in Figure 12.

```
+-------------+         +--------------+
|             |<-(1)--- DOTS signal ------>|              |
|             |         channel session 1  |              |
|             |<=(2)== redirect to B ======|              |
| DOTS client |                            | DOTS server A|
|             |X-(4)--- DOTS signal ------X|              |
|             |         channel session 1  |              |
|             |                            |              |
+-------------+         +--------------+
      ^
      |
   (3) DOTS signal channel
      |       session 2
      v
+--------------+
| DOTS server B |
+--------------+
```
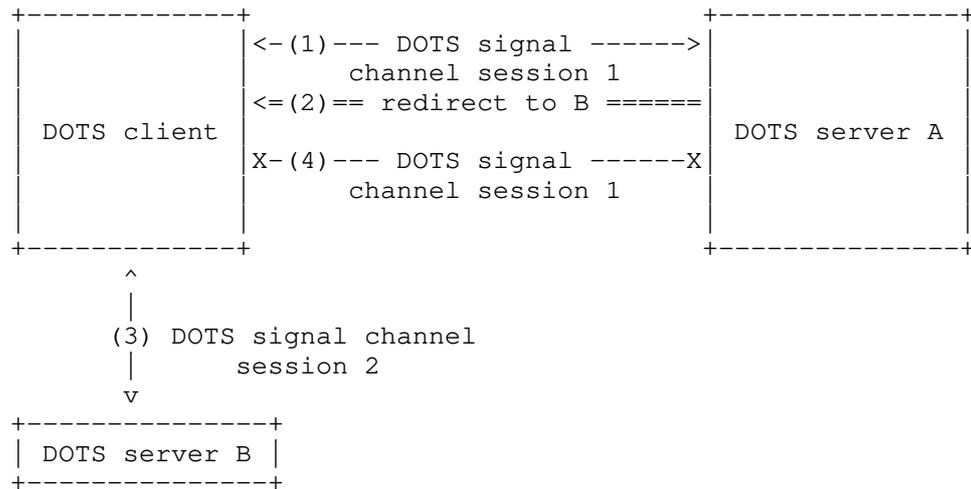
Figure 12: Redirected Signaling

1. Previously established DOTS signal channel session 1 exists
    between a DOTS client and DOTS server A.

2. DOTS server A sends a server signal redirecting the client to
    DOTS server B.

3. If the DOTS client does not already have a separate DOTS signal
    channel session with the redirection target, the DOTS client
    initiates and establishes DOTS signal channel session 2 with DOTS
    server B.

4. Having redirected the DOTS client, DOTS server A ceases sending
    server signals. The DOTS client likewise stops sending client
    signals to DOTS server A. DOTS signal channel session 1 is
    terminated.

3.2.3. Recursive Signaling

   DOTS is centered around improving the speed and efficiency of
   coordinated response to DDoS attacks. One scenario not yet discussed
   involves coordination among federated domains operating DOTS servers
   and mitigators.

   In the course of normal DOTS operations, a DOTS client communicates
   the need for mitigation to a DOTS server, and that server initiates
   mitigation on a mitigator with which the server has an established
   service relationship. The operator of the mitigator may in turn
   monitor mitigation performance and capacity, as the attack being
   mitigated may grow in severity beyond the mitigating domain's
   capabilities.

   The operator of the mitigator has limited options in the event a DOTS
   client-requested mitigation is being overwhelmed by the severity of
   the attack. Out-of-scope business or service level agreements may
   permit the mitigating domain to drop the mitigation and let attack
   traffic flow unchecked to the target, but this only encourages attack
   escalation. In the case where the mitigating domain is the upstream
   service provider for the attack target, this may mean the mitigating
   domain and its other services and users continue to suffer the
   incidental effects of the attack.

   A recursive signaling model as shown in Figure 13 offers an
   alternative. In a variation of the use case "Upstream DDoS
   Mitigation by an Upstream Internet Transit Provider" described in
   [I-D.ietf-dots-use-cases], a domain operating a DOTS server and
   mitigator also operates a DOTS client. This DOTS client has an
   established DOTS session with a DOTS server belonging to a separate
   administrative domain.

   With these preconditions in place, the operator of the mitigator
   being overwhelmed or otherwise performing inadequately may request
   mitigation for the attack target from this separate DOTS-aware
   domain. Such a request recurses the originating mitigation request
   to the secondary DOTS server, in the hope of building a cumulative
   mitigation against the attack.

```
                    example.net domain
                    . . . . . . . . . . . . . . . . . .
                    .     Gn                          .
       +----+    1  .   +----+        +-----------+    .
       | Cc |<-------->| Sn |~~~~~~~|  Mitigator |    .
       +----+       .   +====+        |    Mn     |    .
                    .   | Cn |        +-----------+    .
       example.com  .   +----+                         .
         client     .     ^                            .
                    . . . .|. . . . . . . . . . . . . .
                           |
                      2    |
                           |
                    . . . .|. . . . . . . . . . . . . .
                    .      v                           .
                    .   +----+        +-----------+    .
                    .   | So |~~~~~~~|  Mitigator |    .
                    .   +----+        |    Mo     |    .
                    .                 +-----------+    .
                    .                                  .
                    .                                  .
                    . . . . . . . . . . . . . . . . . .
                    example.org domain
```

Figure 13: Recursive Signaling

In Figure 13, client Cc signals a request for mitigation across
inter-domain DOTS session 1 to the DOTS server Sn belonging to the
example.net domain. DOTS server Sn enables mitigation on mitigator
Mn. DOTS server Sn is half of DOTS gateway Gn, being deployed
logically back-to-back with DOTS client Cn, which has pre-existing
inter-domain DOTS session 2 with the DOTS server So belonging to the
example.org domain. At any point, DOTS server Sn MAY recurse an on-
going mitigation request through DOTS client Cn to DOTS server So, in
the expectation that mitigator Mo will be activated to aid in the
defense of the attack target.

Recursive signaling is opaque to the DOTS client. To maximize
mitigation visibility to the DOTS client, however, the recursing
domain SHOULD provide recursed mitigation feedback in signals
reporting on mitigation status to the DOTS client. For example, the
recursing domain's mitigator should incorporate into mitigation
status messages available metrics such as dropped packet or byte
counts from the recursed mitigation.

DOTS clients involved in recursive signaling must be able to withdraw
requests for mitigation without warning or justification, per SIG-006
in [RFC8612].

Operators recursing mitigation requests MAY maintain the recursed
mitigation for a brief, protocol-defined period in the event the DOTS
client originating the mitigation withdraws its request for help, as
per the discussion of managing mitigation toggling in SIG-006 of
[RFC8612].

Deployment of recursive signaling may result in traffic redirection,
examination and mitigation extending beyond the initial bilateral
relationship between DOTS client and DOTS server. As such, client
control over the network path of mitigated traffic may be reduced.
DOTS client operators should be aware of any privacy concerns, and
work with DOTS server operators employing recursive signaling to
ensure shared sensitive material is suitably protected.

3.2.4. Anycast Signaling

The DOTS architecture does not assume the availability of anycast
within a DOTS deployment, but neither does the architecture exclude
it. Domains operating DOTS servers MAY deploy DOTS servers with an
anycast Service Address as described in BCP 126 [RFC4786]. In such a
deployment, DOTS clients connecting to the DOTS Service Address may
be communicating with distinct DOTS servers, depending on the network
configuration at the time the DOTS clients connect. Among other
benefits, anycast signaling potentially offers the following:

o Simplified DOTS client configuration, including service discovery
   through the methods described in [RFC7094]. In this scenario, the
   "instance discovery" message would be a DOTS client initiating a
   DOTS session to the DOTS server anycast Service Address, to which
   the DOTS server would reply with a redirection to the DOTS server
   unicast address the client should use for DOTS.

o Region- or customer-specific deployments, in which the DOTS
   Service Addresses route to distinct DOTS servers depending on the
   client region or the customer network in which a DOTS client
   resides.

o Operational resiliency, spreading DOTS signaling traffic across
   the DOTS server domain's networks, and thereby also reducing the
   potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

As long as network configuration remains stable, anycast DOTS
signaling is to the individual DOTS client indistinct from direct
signaling. However, the operational challenges inherent in anycast
signaling are anything but negligible, and DOTS server operators must
carefully weigh the risks against the benefits before deploying.

While the DOTS signal channel primarily operates over UDP per SIG-001
in [RFC8612], the signal channel also requires mutual authentication
between DOTS agents, with associated security state on both ends.

Network instability is of particular concern with anycast signaling,
as DOTS signal channels are expected to be long-lived, and
potentially operating under congested network conditions caused by a
volumetric DDoS attack.

For example, a network configuration altering the route to the DOTS
server during active anycast signaling may cause the DOTS client to
send messages to a DOTS server other than the one with which it
initially established a signaling session. That second DOTS server
may not have the security state of the existing session, forcing the
DOTS client to initialize a new DOTS session. This challenge might
in part be mitigated by use of resumption via a PSK in TLS 1.3
[RFC8446] and DTLS 1.3 [I-D.ietf-tls-dtls13] (session resumption in
TLS 1.2 [RFC5246] and DTLS 1.2 [RFC6347]), but keying material must
be available to all DOTS servers sharing the anycast Service Address
in that case.

While the DOTS client will try to establish a new DOTS session with
the DOTS server now acting as the anycast DOTS Service Address, the
link between DOTS client and server may be congested with attack
traffic, making signal session establishment difficult. In such a
scenario, anycast Service Address instability becomes a sort of
signal session flapping, with obvious negative consequences for the
DOTS deployment.

Anycast signaling deployments similarly must also take into account
active mitigations. Active mitigations initiated through a DOTS
session may involve diverting traffic to a scrubbing center. If the
DOTS session flaps due to anycast changes as described above,
mitigation may also flap as the DOTS servers sharing the anycast DOTS
service address toggles mitigation on detecting DOTS session loss,
depending on whether the client has configured mitigation on loss of
signal.

3.2.5. Signaling Considerations for Network Address Translation

Network address translators (NATs) are expected to be a common
feature of DOTS deployments. The Middlebox Traversal Guidelines in
[RFC8085] include general NAT considerations for DOTS deployements
when the signal channel is established over UDP.

Additional DOTS-specific considerations arise when NATs are part of
the DOTS architecture. For example, DDoS attack detection behind a
NAT will detect attacks against internal addresses. A DOTS client

   subsequently asked to request mitigation for the attacked scope of
   addresses cannot reasonably perform the task, due to the lack of
   externally routable addresses in the mitigation scope.

   The following considerations do not cover all possible scenarios, but
   are meant rather to highlight anticipated common issues when
   signaling through NATs.

3.2.5.1. Direct Provisioning of Internal-to-External Address Mappings

   Operators may circumvent the problem of translating internal
   addresses or prefixes to externally routable mitigation scopes by
   directly provisioning the mappings of external addresses to internal
   protected resources on the DOTS client. When the operator requests
   mitigation scoped for internal addresses, directly or through
   automated means, the DOTS client looks up the matching external
   addresses or prefixes, and issues a mitigation request scoped to that
   externally routable information.

   When directly provisioning the address mappings, operators must
   ensure the mappings remain up to date, or risk losing the ability to
   request accurate mitigation scopes. To that aim, the DOTS client can
   rely on mechanisms, such as [RFC8512] to retrieve static explicit
   mappings. This document does not prescribe the method by which
   mappings are maintained once they are provisioned on the DOTS client.

3.2.5.2. Resolving Public Mitigation Scope with Port Control Protocol
          (PCP)

   Port Control Protocol (PCP) [RFC6887] may be used to retrieve the
   external addresses/prefixes and/or port numbers if the NAT function
   embeds a PCP server.

   A DOTS client can use the information retrieved by means of PCP to
   feed the DOTS protocol(s) messages that will be sent to a DOTS
   server. These messages will convey the external addresses/prefixes
   as set by the NAT.

   PCP also enables discovery and configuration of the lifetime of port
   mappings instantiated in intermediate NAT devices. Discovery of port
   mapping lifetimes can reduce the dependency on heartbeat messages to
   maintain mappings, and therefore reduce the load on DOTS servers and
   the network.

3.2.5.3. Resolving Public Mitigation Scope with Session Traversal
          Utilities (STUN)

   An internal resource, e.g., a Web server, can discover its reflexive
   transport address through a STUN Binding request/response
   transaction, as described in [RFC5389]. After learning its reflexive
   transport address from the STUN server, the internal resource can
   export its reflexive transport address and internal transport address
   to the DOTS client, thereby enabling the DOTS client to request
   mitigation with the correct external scope, as depicted in Figure 14.
   The mechanism for providing the DOTS client with the reflexive
   transport address and internal transport address is unspecified in
   this document.

   In order to prevent an attacker from modifying the STUN messages in
   transit, the STUN client and server MUST use the message-integrity
   mechanism discussed in Section 10 of [RFC5389] or use STUN over DTLS
   [RFC7350] or use STUN over TLS. If the STUN client is behind a NAT
   that performs Endpoint-Dependent Mapping [RFC5128], the internal
   service cannot provide the DOTS client with the reflexive transport
   address discovered using STUN. The behavior of a NAT between the
   STUN client and the STUN server could be discovered using the
   experimental techniques discussed in [RFC5780], but note that there
   is currently no standardized way for a STUN client to reliably
   determine if it is behind a NAT that performs Endpoint-Dependent
   Mapping.

```
              Binding            Binding
 +--------+   request   +---+    request   +--------+
 |  STUN  |<----------- | N |<----------   |  STUN  |
 | server |             | A |              | client |
 |        |----------> | T | ---------->   |        |
 +--------+   Binding   +---+   Binding    +--------+
              response          response   |
                                           |
                                           | reflexive transport address
                                           | & internal transport address
                                           v
                                    +--------+
                                    |  DOTS  |
                                    | client |
                                    +--------+
```
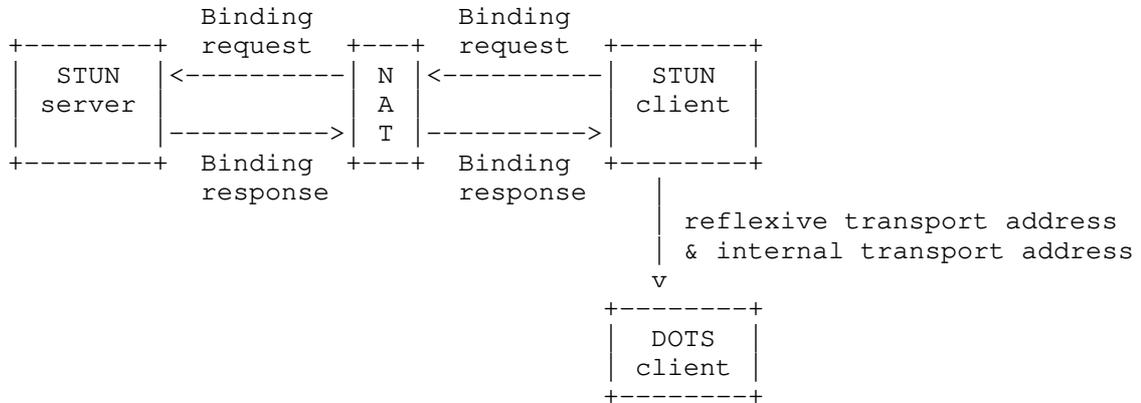
              Figure 14: Resolving mitigation scope with STUN

3.2.5.4. Resolving Requested Mitigation Scope with DNS

   DOTS supports mitigation scoped to DNS names. As discussed in
   [RFC3235], using DNS names instead of IP addresses potentially avoids
   the address translation problem, as long as the name is internally
   and externally resolvable by the same name. For example, a detected
   attack's internal target address can be mapped to a DNS name through
   a reverse lookup. The DNS name returned by the reverse lookup can
   then be provided to the DOTS client as the external scope for
   mitigation. For the reverse DNS lookup, DNS Security Extensions
   (DNSSEC) [RFC4033] must be used where the authenticity of response is
   critical.

3.3. Triggering Requests for Mitigation

   [RFC8612] places no limitation on the circumstances in which a DOTS
   client operator may request mitigation, nor does it demand
   justification for any mitigation request, thereby reserving
   operational control over DDoS defense for the domain requesting
   mitigation. This architecture likewise does not prescribe the
   network conditions and mechanisms triggering a mitigation request
   from a DOTS client.

   However, considering selected possible mitigation triggers from an
   architectural perspective offers a model for alternative or
   unanticipated triggers for DOTS deployments. In all cases, what
   network conditions merit a mitigation request are at the discretion
   of the DOTS client operator.

   The mitigation request itself is defined by DOTS, however the
   interfaces required to trigger the mitigation request in the
   following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

   A DOTS client operator may manually prepare a request for mitigation,
   including scope and duration, and manually instruct the DOTS client
   to send the mitigation request to the DOTS server. In context, a
   manual request is a request directly issued by the operator without
   automated decision-making performed by a device interacting with the
   DOTS client. Modes of manual mitigation requests include an operator
   entering a command into a text interface, or directly interacting
   with a graphical interface to send the request.

   An operator might do this, for example, in response to notice of an
   attack delivered by attack detection equipment or software, and the
   alerting detector lacks interfaces or is not configured to use

available interfaces to translate the alert to a mitigation request
automatically.

In a variation of the above scenario, the operator may have
preconfigured on the DOTS client mitigation requests for various
resources in the operator's domain. When notified of an attack, the
DOTS client operator manually instructs the DOTS client to send the
relevant preconfigured mitigation request for the resources under
attack.

A further variant involves recursive signaling, as described in
Section 3.2.3. The DOTS client in this case is the second half of a
DOTS gateway (back-to-back DOTS server and client). As in the
previous scenario, the scope and duration of the mitigation request
are pre-existing, but in this case are derived from the mitigation
request received from a downstream DOTS client by the DOTS server.
Assuming the preconditions required by Section 3.2.3 are in place,
the DOTS gateway operator may at any time manually request mitigation
from an upstream DOTS server, sending a mitigation request derived
from the downstream DOTS client's request.

The motivations for a DOTS client operator to request mitigation
manually are not prescribed by this architecture, but are expected to
include some of the following:

o Notice of an attack delivered via e-mail or alternative messaging

o Notice of an attack delivered via phone call

o Notice of an attack delivered through the interface(s) of
   networking monitoring software deployed in the operator's domain

o Manual monitoring of network behavior through network monitoring
   software

3.3.2. Automated Conditional Mitigation Request

Unlike manual mitigation requests, which depend entirely on the DOTS
client operator's capacity to react with speed and accuracy to every
detected or detectable attack, mitigation requests triggered by
detected attack conditions reduce the operational burden on the DOTS
client operator, and minimize the latency between attack detection
and the start of mitigation.

Mitigation requests are triggered in this scenario by operator-
specified network conditions. Attack detection is deployment-
specific, and not constrained by this architecture. Similarly the

specifics of a condition are left to the discretion of the operator,
though common conditions meriting mitigation include the following:

o Detected attack exceeding a rate in packets per second (pps).

o Detected attack exceeding a rate in bytes per second (bps).

o Detected resource exhaustion in an attack target.

o Detected resource exhaustion in the local domain's mitigator.

o Number of open connections to an attack target.

o Number of attack sources in a given attack.

o Number of active attacks against targets in the operator's domain.

o Conditional detection developed through arbitrary statistical
   analysis or deep learning techniques.

o Any combination of the above.

When automated conditional mitigation requests are enabled,
violations of any of the above conditions, or any additional
operator-defined conditions, will trigger a mitigation request from
the DOTS client to the DOTS server. The interfaces between the
application detecting the condition violation and the DOTS client are
implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

To maintain a DOTS signal channel session, the DOTS client and the
DOTS server exchange regular but infrequent messages across the
signal channel. In the absence of an attack, the probability of
message loss in the signaling channel should be extremely low. Under
attack conditions, however, some signal loss may be anticipated as
attack traffic congests the link, depending on the attack type.

While [RFC8612] specifies the DOTS protocol be robust when signaling
under attack conditions, there are nevertheless scenarios in which
the DOTS signal is lost in spite of protocol best efforts. To handle
such scenarios, a DOTS operator may request one or more mitigations
which are triggered only when the DOTS server ceases receiving DOTS
client heartbeats beyond the miss count or interval permitted by the
protocol.

The impact of mitigating due to loss of signal in either direction
must be considered carefully before enabling it. Signal loss is not

caused by links congested with attack traffic alone, and as such
mitigation requests triggered by signal channel degradation in either
direction may incur unnecessary costs, in network performance and
operational expense alike.

4. IANA Considerations

   This document has no actions for IANA.

5. Security Considerations

   This section describes identified security considerations for the
   DOTS architecture.

   DOTS is at risk from three primary attack vectors: agent
   impersonation, traffic injection and signal blocking. These vectors
   may be exploited individually or in concert by an attacker to
   confuse, disable, take information from, or otherwise inhibit DOTS
   agents.

   Any attacker with the ability to impersonate a legitimate DOTS client
   or server or, indeed, inject false messages into the stream may
   potentially trigger/withdraw traffic redirection, trigger/cancel
   mitigation activities or subvert drop-/accept-lists. From an
   architectural standpoint, operators SHOULD ensure best current
   practices for secure communication are observed for data and signal
   channel confidentiality, integrity and authenticity. Care must be
   taken to ensure transmission is protected by appropriately secure
   means, reducing attack surface by exposing only the minimal required
   services or interfaces. Similarly, received data at rest SHOULD be
   stored with a satisfactory degree of security.

   As many mitigation systems employ diversion to scrub attack traffic,
   operators of DOTS agents SHOULD ensure DOTS sessions are resistant to
   Man-in-the-Middle (MitM) attacks. An attacker with control of a DOTS
   client may negatively influence network traffic by requesting and
   withdrawing requests for mitigation for particular prefixes, leading
   to route or DNS flapping.

   Any attack targeting the availability of DOTS servers may disrupt the
   ability of the system to receive and process DOTS signals resulting
   in failure to fulfill a mitigation request. DOTS agents SHOULD be
   given adequate protections, again in accordance with best current
   practices for network and host security.

6. Contributors

   Mohamed Boucadair
      Orange

      mohamed.boucadair@orange.com

   Christopher Gray   Christopher_Gray3@cable.comcast.com

7. Acknowledgments

   Thanks to Matt Richardson, Roman Danyliw, Frank Xialiang, Roland
   Dobbins, Wei Pan, Kaname Nishizuka, Jon Shallow, and Mohamed
   Boucadair for their comments and suggestions.

8. References

8.1. Normative References

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

   [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

   [I-D.ietf-dots-use-cases]
               Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
               Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
               Open Threat Signaling", draft-ietf-dots-use-cases-17 (work
               in progress), January 2019.

   [I-D.ietf-tls-dtls13]
               Rescorla, E., Tschofenig, H., and N. Modadugu, "The
               Datagram Transport Layer Security (DTLS) Protocol Version
               1.3", draft-ietf-tls-dtls13-31 (work in progress), March
               2019.

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

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

   [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>.

   [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
              specifying the location of services (DNS SRV)", RFC 2782,
              DOI 10.17487/RFC2782, February 2000,
              <https://www.rfc-editor.org/info/rfc2782>.

   [RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly
              Application Design Guidelines", RFC 3235,
              DOI 10.17487/RFC3235, January 2002,
              <https://www.rfc-editor.org/info/rfc3235>.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [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>.

   [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A
              Border Gateway Protocol 4 (BGP-4)", RFC 4271,
              DOI 10.17487/RFC4271, January 2006,
              <https://www.rfc-editor.org/info/rfc4271>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [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>.

   [RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-
              Peer (P2P) Communication across Network Address
              Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March
              2008, <https://www.rfc-editor.org/info/rfc5128>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              DOI 10.17487/RFC5389, October 2008,
              <https://www.rfc-editor.org/info/rfc5389>.

   [RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery
              Using Session Traversal Utilities for NAT (STUN)",
              RFC 5780, DOI 10.17487/RFC5780, May 2010,
              <https://www.rfc-editor.org/info/rfc5780>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
              DOI 10.17487/RFC6887, April 2013,
              <https://www.rfc-editor.org/info/rfc6887>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <https://www.rfc-editor.org/info/rfc7092>.

   [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
              "Architectural Considerations of IP Anycast", RFC 7094,
              DOI 10.17487/RFC7094, January 2014,
              <https://www.rfc-editor.org/info/rfc7094>.

   [RFC7350] Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport
              Layer Security (DTLS) as Transport for Session Traversal
              Utilities for NAT (STUN)", RFC 7350, DOI 10.17487/RFC7350,
              August 2014, <https://www.rfc-editor.org/info/rfc7350>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

   [RFC8446]   Rescorla, E., "The Transport Layer Security (TLS) Protocol
               Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
               <https://www.rfc-editor.org/info/rfc8446>.

   [RFC8512]   Boucadair, M., Ed., Sivakumar, S., Jacquenet, C.,
               Vinapamula, S., and Q. Wu, "A YANG Module for Network
               Address Translation (NAT) and Network Prefix Translation
               (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019,
               <https://www.rfc-editor.org/info/rfc8512>.

   [RFC8612]   Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
               Threat Signaling (DOTS) Requirements", RFC 8612,
               DOI 10.17487/RFC8612, May 2019,
               <https://www.rfc-editor.org/info/rfc8612>.

Authors' Addresses

   Andrew Mortensen (editor)
   Forcepoint
   United States

   EMail: andrewmortensen@gmail.com

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

   EMail: kondtir@gmail.com

   Flemming Andreasen
   Cisco
   United States

   EMail: fandreas@cisco.com

   Nik Teague
   Iron Mountain
   United States

   EMail: nteague@ironmountain.co.uk

Rich Compton
Charter

EMail: Rich.Compton@charter.com

DOTS   T. Reddy
Internet-Draft   McAfee
Intended status: Standards Track   M. Boucadair
Expires: May 16, 2018   Orange
                                                          K. Nishizuka
                                                    NTT Communications
                                                               L. Xia
                                                               Huawei
                                                             P. Patil
                                                                Cisco
                                                          A. Mortensen
                                                  Arbor Networks, Inc.
                                                             N. Teague
                                                         Verisign, Inc.
                                                     November 12, 2017

          Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel
                       draft-ietf-dots-data-channel-07

   Abstract

      The document specifies a Distributed Denial-of-Service Open Threat
      Signaling (DOTS) data channel used for bulk exchange of data not
      easily or appropriately communicated through the DOTS signal channel
      under attack conditions. This is a companion document to the DOTS
      signal channel specification.

Table of Contents

1. Introduction

   A distributed denial-of-service (DDoS) attack is an attempt to make
   machines or network resources unavailable to their intended users.
   In most cases, sufficient scale can be achieved by compromising
   enough end-hosts and using those infected hosts to perpetrate and
   amplify the attack. The victim in this attack can be an application
   server, a client, a router, a firewall, or an entire network.

   DDoS Open Threat Signaling (DOTS) [I-D.ietf-dots-architecture]
   defines two channels: signal and data channels (Figure 1). The DOTS
   signal channel used to convey that a network is under a DDOS attack
   to an upstream DOTS server so that appropriate mitigation actions are
   undertaken on the suspect traffic is further elaborated in
   [I-D.ietf-dots-signal-channel]. The DOTS data channel is used for
   infrequent bulk data exchange between DOTS agents in the aim to
   significantly augment attack response coordination.

   ```
   +--------------+                              +--------------+
   |              | <------- Signal Channel ------> |              |
   |  DOTS Client |                              | DOTS Server  |
   |              | <======= Data Channel  ======> |              |
   +--------------+                              +--------------+
   ```
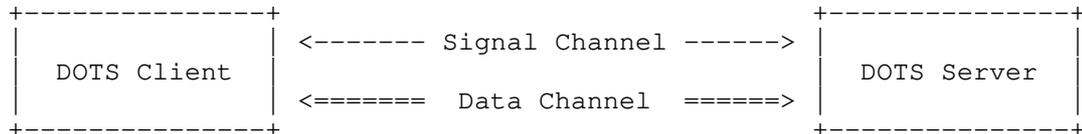
                      Figure 1: DOTS Channels

   Section 2 of [I-D.ietf-dots-architecture] identifies that the DOTS
   data channel is used to perform the tasks listed below:

   o Filter management, which enables a DOTS client to request the
      installation or removal of traffic filters, dropping or rate-
      limiting unwanted traffic and permitting white-listed traffic.
      Sample use cases for populating black- or white-list filtering
      rules are detailed hereafter:

      A. If a network resource (DOTS client) detects a potential DDoS
          attack from a set of IP addresses, the DOTS client informs its
          servicing router (DOTS gateway) of all suspect IP addresses
          that need to be blocked or black-listed for further
          investigation. The DOTS client could also specify a list of
          protocols and ports in the black-list rule. That DOTS gateway
          in-turn propagates the black-listed IP addresses to the DOTS
          server which will undertake appropriate action so that traffic
          from these IP addresses to the target network (specified by
          the DOTS client) is blocked.

      B. A network has partner sites from which only legitimate traffic
          arrives and the network wants to ensure that the traffic from

these sites is not penalized during DDOS attacks. The DOTS
client uses the DOTS data channel to convey the white-listed
IP addresses or prefixes of the partner sites to its DOTS
server. The DOTS server uses this information to white-list
flows from such IP addresses or prefixes reaching the network.

o Creating identifiers, such as names or aliases, for resources for
which mitigation may be requested:

A. The DOTS client may submit to the DOTS server a collection of
prefixes which it would like to refer to by alias when
requesting mitigation. The server can respond to this request
with either with a success or failure response (see
requirement OP-006 in [I-D.ietf-dots-requirements] and
Section 2 in [I-D.ietf-dots-architecture]).

2. Notational Conventions and Terminology

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

The reader should be familiar with the terms defined in
[I-D.ietf-dots-architecture].

The terminology for describing YANG data modules is defined in
[RFC7950]. The meaning of the symbols in tree diagrams is defined in
[I-D.ietf-netmod-yang-tree-diagrams].

For simplicity, all of the examples in this document use "/restconf"
as the discovered RESTCONF API root path. Many protocol header lines
and message-body text within examples throughout the document are
split into multiple lines for display purposes only. When a line
ends with backslash ('\') as the last character, the line is wrapped
for display purposes. It is to be considered to be joined to the
next line by deleting the backslash, the following line break, and
the leading whitespace of the next line.

3. DOTS Data Channel

The DOTS data channel is intended to be used for bulk data exchanges
between DOTS agents. Unlike the signal channel
[I-D.ietf-dots-signal-channel], which must operate nominally even
when confronted with signal degradation due to packets loss, the data
channel is not expected to be constructed to deal with DDoS attack
conditions.

As the primary function of the data channel is data exchange, a
reliable transport is required in order for DOTS agents to detect
data delivery success or failure. RESTCONF [RFC8040] over TLS
[RFC5246] over TCP is used for DOTS data channel (Figure 2).
RESTCONF uses HTTP methods to provide CRUD (create, read, update,
delete) operations on a conceptual datastore containing YANG data,
which is compatible with a server implementing NETCONF datastores.

The HTTP POST, PUT, PATCH, and DELETE methods are used to edit data
resources represented by DOTS data channel YANG data models. These
basic edit operations allow the DOTS data channel running
configuration to be altered by a DOTS client.

DOTS data channel configuration data and state data can be retrieved
with the GET method. HTTP status codes are used to report success or
failure for RESTCONF operations.

The DOTS client will perform the root resource discovery procedure
discussed in Section 3.1 of [RFC8040] to determine the root of the
RESTCONF API. After discovering the RESTCONF API root, the DOTS
client uses this value as the initial part of the path in the request
URI, in any subsequent request to the DOTS server. The DOTS server
may support retrieval of the YANG modules it supports (Section 3.7 in
[RFC8040]), for example, a DOTS client may use RESTCONF to retrieve
the company proprietary YANG modules supported by the DOTS server.

   Note: This document uses RESTCONF, a protocol based on HTTP
   [RFC7230], for configuring data defined in YANG version 1
   [RFC6020] or YANG version 1.1 [RFC7950], using the datastore
   concepts defined in the Network Configuration Protocol (NETCONF)
   [RFC6241]. RESTCONF combines the simplicity of the HTTP protocol
   with the predictability and automation potential of a schema-
   driven API. RESTCONF offers a simple subset of NETCONF
   functionality and provides a simplified interface using REST-like
   API which addresses the needs of the DOTS data channel and hence
   an optimal choice.

```
                    +--------------+
                    |     DOTS     |
                    +--------------+
                    |   RESTCONF   |
                    +--------------+
                    |     TLS      |
                    +--------------+
                    |     TCP      |
                    +--------------+
                    |      IP      |
                    +--------------+
```
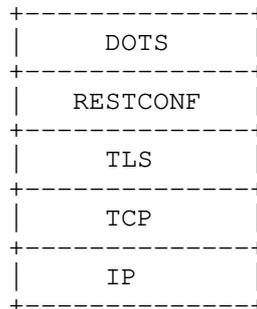
Figure 2: Abstract Layering of DOTS data channel over RESTCONF over TLS

JavaScript Object Notation (JSON) [RFC7159] payload is used to propagate data channel specific payload messages that convey request parameters and response information such as errors. This specification uses the encoding rules defined in [RFC7951] for representing DOTS data channel configuration data defined using YANG (Section 3.1) as JSON text.

A DOTS client registers itself to its DOTS server(s) in order to set up DOTS data channel related configuration data and receive state data (i.e., non-configuration data) from the DOTS server(s).

A single DOTS data channel between DOTS agents can be used to exchange multiple requests and multiple responses. To reduce DOTS client and DOTS server workload, DOTS client SHOULD re-use the same TLS session. While the communication to the DOTS server is quiescent, the DOTS client MAY probe the server to ensure it has maintained cryptographic state. Such probes can also keep alive firewall and/or NAT bindings. A TLS heartbeat [RFC6520] verifies the DOTS server still has TLS state by returning a TLS message.

3.1. DOTS Data Channel YANG Module

3.1.1. Identifier Module Tree Structure

This document defines a YANG module for creating identifiers, such as names or aliases, for resources for which mitigation may be requested. Such identifiers may be used in subsequent DOTS signal channel exchanges to refer more efficiently to the resources under attack.

This document defines the YANG module "ietf-dots-data-channel-identifier", which has the following tree structure:

```
module: ietf-dots-data-channel-identifier
    +--rw identifier
       +--rw client-identifier*   binary
       +--rw alias* [alias-name]
          +--rw alias-name           string
          +--rw target-ip*           inet:ip-address
          +--rw target-prefix*       inet:ip-prefix
          +--rw target-port-range* [lower-port upper-port]
          │  +--rw lower-port    inet:port-number
          │  +--rw upper-port    inet:port-number
          +--rw target-protocol*     uint8
          +--rw fqdn*                inet:domain-name
          +--rw uri*                 inet:uri
```

This structure is aligned with Section 5.2.1 of
[I-D.ietf-dots-signal-channel].

3.1.2. Identifier Model YANG Module

<CODE BEGINS> file "ietf-dots-data-channel-identifier@2017-10-12.yang"

```
module ietf-dots-data-channel-identifier {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-dots-data-channel-identifier";

     prefix "alias";

     import ietf-inet-types {
         prefix "inet";
     }

   organization "IETF DOTS Working Group";

   contact
     "Konda, Tirumaleswar Reddy <TirumaleswarReddy_Konda@McAfee.com>
      Mohamed Boucadair <mohamed.boucadair@orange.com>
      Kaname Nishizuka <kaname@nttv6.jp>
      Liang Xia <frank.xialiang@huawei.com>
      Prashanth Patil <praspati@cisco.com>
      Andrew Mortensen <amortensen@arbor.net>
      Nik Teague <nteague@verisign.com>";

    description
      "This module contains YANG definition for configuring
       identifiers for resources using DOTS data channel.

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.
```

      revision 2017-10-12 {
        description "Fix nits and align the module with the signal
                    channel.";
        reference
        "-05";
      }

      revision 2017-08-03 {
        reference
        "https://tools.ietf.org/html/draft-ietf-dots-data-channel";
      }

      container identifier {
          description "Top level container for identifiers";

              leaf-list client-identifier {
                 type binary;
                 description "A client identifier conveyed by a DOTS gateway
                              to a remote DOTS server.";

                 reference
                   "I-D.itef-dots-signal-channel";
              }

              list alias {
                  key alias-name;
                  description "List of identifiers";

                  leaf alias-name {
                     type string;
                     description "alias name";
                  }

                  leaf-list target-ip {
                     type inet:ip-address;
                     description "IPv4 or IPv6 address identifying the target."
;
                  }

```
                    leaf-list target-prefix {
                       type inet:ip-prefix;
                       description "IPv4 or IPv6 prefix identifying the target.";
                    }

                    list target-port-range {
                       key "lower-port upper-port";
                       description
                           "Port range. When only lower-port is present,
                            it represents a single port.";
                       leaf lower-port {
                          type inet:port-number;
                          mandatory true;
                          description "Lower port number.";
                       }
                       leaf upper-port {
                          type inet:port-number;
                          must ". >= ../lower-port" {
                            error-message
                            "The upper-port must be greater than or
                             equal to lower-port";
                          }
                          description "Upper port number.";
                       }
                    }

                    leaf-list target-protocol {
                       type uint8;
                       description "Identifies the target protocol number.";

                       reference
                        "https://www.iana.org/assignments/protocol-numbers/
                         protocol-numbers.xhtml";
                    }

                    leaf-list fqdn {
                      type inet:domain-name;
                      description "FQDN";
                    }

                    leaf-list uri {
                      type inet:uri;
                      description "URI";
                    }
                }
            }
        }
    <CODE ENDS>
```

3.1.3. Filter Model YANG Module Tree Structure

   This document augments the "ietf-access-control-list" Access Control
   List (ACL) YANG module [I-D.ietf-netmod-acl-model] for managing
   filtering rules. ACL is explained in Section 1 of
   [I-D.ietf-netmod-acl-model].

   Examples of ACL management include, but not limited to,:

   o Black-list management, which enables a DOTS client to inform the
      DOTS server about sources from which traffic should be suppressed.

   o White-list management, which enables a DOTS client to inform the
      DOTS server about sources from which traffic should always be
      accepted.

   o Filter management, which enables a DOTS client to request the
      installation or removal of traffic filters, dropping or rate-
      limiting unwanted traffic and permitting white-listed traffic.

   This document defines the YANG module "ietf-dots-access-control-list"
   to augment the "ietf-access-control-list" module to support filters
   based on the client identifier (client-identifier), to support rate-
   limit action (rate-limit), and to handle fragmented packets
   (fragments).

   Filtering fragments adds an additional layer of protection against a
   DoS attack that uses only noninitial fragments. When there is only
   Layer 3 information in the ACL entry and the fragments keyword is
   present, for noninitial fragments matching the ACL entry, the deny or
   permit action associated with the ACL entry will be enforced and for
   initial or non-fragment matching the ACL entry, the next ACL entry
   will be processed. When there is both Layer 3 and Layer 4
   information in the ACL entry and the fragments keyword is present,
   the ACL action is conservative for both permit and deny actions. The
   actions are conservative to not accidentally deny a fragmented
   portion of a flow because the fragments do not contain sufficient
   information to match all of the filter attributes. In the deny
   action case, instead of denying a non-initial fragment, the next ACL
   entry is processed. In the permit case, it is assumed that the Layer
   4 information in the non-initial fragment, if available, matches the
   Layer 4 information in the ACL entry.

   The "ietf-dots-access-control-list" module has the following
   structure:

```
module: ietf-dots-access-control-list
  augment /ietf-acl:access-lists:
    +--rw client-identifier*   binary
  augment /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ace/ietf-ac
l:actions/ietf-acl:packet-handling:
    +--:(rate-limit)
       +--rw rate-limit?   decimal64
  augment /ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ace:
    +--rw fragments?   empty
```

3.1.4. Filter Model YANG Module

<CODE BEGINS> file "ietf-dots-access-control-list@2017-10-12.yang"

```
module ietf-dots-access-control-list {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-dots-access-control-list";
    prefix "dots-acl";

    import ietf-access-control-list {
     prefix "ietf-acl";
    }

  organization "IETF DOTS Working Group";

  contact
    "Konda, Tirumaleswar Reddy <TirumaleswarReddy_Konda@McAfee.com>
     Mohamed Boucadair <mohamed.boucadair@orange.com>
     Kaname Nishizuka <kaname@nttv6.jp>
     Liang Xia <frank.xialiang@huawei.com>
     Prashanth Patil <praspati@cisco.com>
     Andrew Mortensen <amortensen@arbor.net>
     Nik Teague <nteague@verisign.com>";

   description
     "This module contains YANG definition for configuring
      filtering rules using DOTS data channel.

      Copyright (c) 2017 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).
```

```
         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";

    revision 2017-10-12 {
      description "Fix nits and align the module with the signal
                  channel.";
      reference
      "-05";
    }

     revision 2017-06-12 {
      reference
      "https://tools.ietf.org/html/draft-ietf-dots-data-channel";
     }

   augment "/ietf-acl:access-lists" {

     description "client-identifier parameter.";

     leaf-list client-identifier {
         type binary;
         description  "A client identifier conveyed by a DOTS gateway
                       to a remote DOTS server.";
     }
   }

   augment "/ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/"
           + "ietf-acl:ace/ietf-acl:actions" {

     description "rate-limit action";
      leaf rate-limit {
         type decimal64 {
          fraction-digits 2;
         }
         description "rate-limit action.";
       }
   }

   augment "/ietf-acl:access-lists/ietf-acl:acl/ietf-acl:aces/ietf-acl:ace" {

     description "Handle non-initial and initial fragments.";

     leaf fragments {
         type empty;
         description "Handle fragments.";
     }
   }
}
```

 <CODE ENDS>

3.2. Identifiers

3.2.1. Create Identifiers

   A POST request is used to create identifiers, such as names or
   aliases, for resources for which a mitigation may be requested. Such
   identifiers may then be used in subsequent DOTS signal channel
   exchanges to refer more efficiently to the resources under attack
   (Figure 3).

```
   POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
   Host: {host}:{port}
   Content-Format: "application/yang.api+json"
   {
    "ietf-dots-data-channel-identifier:identifier": {
      "client-identifier": [
          "string"
      ],
      "alias": [
        {
          "alias-name": "string",
          "target-ip": [
            "string"
          ],
          "target-prefix": [
            "string"
          ],
          "target-port-range": [
            {
              "lower-port": integer,
              "upper-port": integer
            }
          ],
          "target-protocol": [
            integer
          ],
          "fqdn": [
            "string"
          ],
          "uri": [
            "string"
          ]
        }
      ]
    }
   }
```

                    Figure 3: POST to create identifiers

   The header parameters are described below:

   client-identifer: This attribute has the same meaning, syntax and
      processing rules as the 'client-identifier' attribute defined in
      [I-D.ietf-dots-signal-channel]. This is an optional attribute.

   alias-name: Name of the alias. This is a mandatory attribute.

target-ip: IP addresses are separated by commas. This is an
   optional attribute.

target-prefix: Prefixes are separated by commas. This is an
   optional attribute.

target-port-range: The port range, lower-port for lower port number
   and upper-port for upper port number. For TCP, UDP, SCTP, or
   DCCP: the range of ports (e.g., 80 to 8080). This is an optional
   attribute.

target-protocol: Values are taken from the IANA protocol registry
   [proto_numbers]. The value 0 has a special meaning for 'all
   protocols'. This is an optional attribute.

fqdn: Fully Qualified Domain Name, is the full name of a system,
   rather than just its hostname. For example, "venera" is a
   hostname, and "venera.isi.edu" is an FQDN. This is an optional
   attribute.

uri: Uniform Resource Identifier (URI). This is an optional
   attribute.

In the POST request at least one of the attributes 'target-ip' or
'target-prefix' or 'fqdn' or 'uri' MUST be present. DOTS agents can
safely ignore Vendor-Specific parameters they don't understand.

Figure 4 shows a POST request to create alias called "https1" for
HTTP(S) servers with IP addresses 2001:db8:6401::1 and
2001:db8:6401::2 listening on port 443.

```
POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
  "ietf-dots-data-channel-identifier:identifier": {
    "client-identifier": [
      "dz6pHjaADkaFTbjr0JGBpw",
      "iAYmCNPmrYoKoqzgFMiobw"
    ],
    "alias": [
      {
        "alias-name": "Server1",
        "target-protocol": [
          6
        ],
        "target-ip": [
          "2001:db8:6401::1",
          "2001:db8:6401::2"
        ],
        "target-port-range": [
          {
            "lower-port": 443
          }
        ]
      }
    ]
  }
}
```

                  Figure 4: POST to create identifiers

   The DOTS server indicates the result of processing the POST request
   using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx
   codes are some sort of invalid requests and 5xx codes are returned if
   the DOTS server has erred or it is incapable of accepting the alias.
   Response code 201 (Created) will be returned in the response if the
   DOTS server has accepted the alias. If the request is missing one or
   more mandatory attributes then 400 (Bad Request) will be returned in
   the response or if the request contains invalid or unknown parameters
   then 400 (Invalid query) will be returned in the response. The HTTP
   response will include the JSON body received in the request.

   The DOTS client can use the PUT request (Section 4.5 in [RFC8040]) to
   create or modify the aliases in the DOTS server.

3.2.2. Delete Identifiers

   A DELETE request is used to delete identifiers maintained by a DOTS
   server (Figure 5).

```
DELETE /restconf/data/ietf-dots-data-channel-identifier:identifier\
      /client-identifier=dz6pHjaADkaFTbjr0JGBpw,iAYmCNPmrYoKoqzgFMiobw\
      /alias-name=Server1 HTTP/1.1
Host: {host}:{port}
```

                    Figure 5: DELETE identifier

   In RESTCONF, URI-encoded path expressions are used. A RESTCONF data
   resource identifier is encoded from left to right, starting with the
   top-level data node, according to the 'api-path' rule defined in
   Section 3.5.3.1 of [RFC8040]. The data node in the above path
   expression is a YANG list node and MUST be encoded according to the
   rules defined in Section 3.5.1 of [RFC8040].

   If the DOTS server does not find the alias name conveyed in the
   DELETE request in its configuration data, then it responds with a 404
   (Not Found) error response code. The DOTS server successfully
   acknowledges a DOTS client's request to remove the identifier using
   204 (No Content) in the response.

3.2.3. Retrieving Installed Identifiers

   A GET request is used to retrieve the set of installed identifiers
   from a DOTS server (Section 3.3.1 in [RFC8040]). Figure 6 shows how
   to retrieve all the identifiers that were instantiated by the DOTS
   client. The content parameter and its permitted values are defined
   in Section 4.8.1 of [RFC8040].

```
GET /restconf/data/ietf-dots-data-channel-identifier:identifier\
    /client-identifier=dz6pHjaADkaFTbjr0JGBpw,iAYmCNPmrYoKoqzgFMiobw?\
    content=config HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

          Figure 6: GET to retrieve all the installed identifiers

   Figure 7 shows response for all identifiers on the DOTS server.

```
   {
    "ietf-dots-data-channel-identifier:identifier": {
       "client-identifier": [
          "dz6pHjaADkaFTbjr0JGBpw",
          "iAYmCNPmrYoKoqzgFMiobw"
       ],
       "alias": [
          {
             "alias-name": "Server1",
             "traffic-protocol": [
                6
             ],
             "ip": [
                "2001:db8:6401::1",
                "2001:db8:6401::2"
             ],
             "port-range": [
                {
                   "lower-port": 443
                }
             ]
          },
          {
             "alias-name": "Server2",
             "traffic-protocol": [
                6
             ],
             "ip": [
                "2001:db8:6401::10",
                "2001:db8:6401::20"
             ],
             "port-range": [
                {
                   "lower-port": 80
                }
             ]
          }
       ]
    }
   }
```

                        Figure 7: Response body

   If the DOTS server does not find the alias name conveyed in the GET
   request in its configuration data, then it responds with a 404 (Not
   Found) error response code.

3.3. Filtering Rules

   The DOTS server either receives the filtering rules directly from the
   DOTS client or via a DOTS gateway.

   If the DOTS client signals the filtering rules via a DOTS gateway,
   then the DOTS gateway validates if the DOTS client is authorized to
   signal the filtering rules and if the client is authorized propagates
   the rules to the DOTS server. Likewise, the DOTS server validates if
   the DOTS gateway is authorized to signal the filtering rules. To
   create or purge filters, the DOTS client sends HTTP requests to its
   DOTS gateway. The DOTS gateway validates the rules in the requests
   and proxies the requests containing the filtering rules to a DOTS
   server. When the DOTS gateway receives the associated HTTP response
   from the DOTS server, it propagates the response back to the DOTS
   client.

   The following APIs define means for a DOTS client to configure
   filtering rules on a DOTS server.

3.3.1. Install Filtering Rules

   A POST request is used to push filtering rules to a DOTS server.
   Figure 8 shows a POST request example to block traffic from
   192.0.2.0/24, destined to 198.51.100.0/24. The ACL JSON
   configuration for the filtering rule is generated using the ACL YANG
   data model defined in [I-D.ietf-netmod-acl-model] and the ACL
   configuration XML for the filtering rule is specified in Section 4.3
   of [I-D.ietf-netmod-acl-model].

```
POST /restconf/data/ietf-dots-access-control-list HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
 "ietf-dots-access-control-list:access-lists": {
    "client-identifier": [
     "dz6pHjaADkaFTbjr0JGBpw",
     "iAYmCNPmrYoKoqzgFMiobw"
    ],
    "acl": [
        {
            "acl-name": "sample-ipv4-acl",
            "acl-type": "ipv4-acl",
            "aces": {
                "ace": [
                    {
                        "rule-name": "rule1",
                        "matches": {
                          "ipv4-acl": {
                            "source-ipv4-network": "192.0.2.0/24",
                            "destination-ipv4-network": "198.51.100.0/24"
                          }
                        },
                        "actions": {
                            "deny": [null]
                        }
                    }
                ]
            }
        }
    ]
 }
}
```

                  Figure 8: POST to install filterng rules

   The header parameters defined in [I-D.ietf-netmod-acl-model] are
   discussed below:

   acl-name: The name of access-list. This is a mandatory attribute.

   acl-type: Indicates the primary intended type of match criteria
      (e.g. IPv4, IPv6). This is a mandatory attribute.

   protocol: Internet Protocol numbers. This is an optional
      attribute.

source-ipv4-network: The source IPv4 prefix. This is an optional
   attribute.

destination-ipv4-network: The destination IPv4 prefix. This is an
   optional attribute.

actions: "deny" or "permit" or "rate-limit". "permit" action is
   used to white-list traffic. "deny" action is used to black-list
   traffic. "rate-limit" action is used to rate-limit traffic, the
   allowed traffic rate is represented in bytes per second indicated
   in IEEE floating point format [IEEE.754.1985]. If actions
   attribute is not specified in the request then the default action
   is "deny". This is an optional attribute.

The DOTS server indicates the result of processing the POST request
using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx
codes are some sort of invalid requests and 5xx codes are returned if
the DOTS server has erred or it is incapable of configuring the
filtering rules. Response code 201 (Created) will be returned in the
response if the DOTS server has accepted the filtering rules. If the
request is missing one or more mandatory attributes then 400 (Bad
Request) will be returned in the response or if the request contains
invalid or unknown parameters then 400 (Invalid query) will be
returned in the response.

The DOTS client can use the PUT request to create or modify the
filtering rules in the DOTS server.

3.3.2. Remove Filtering Rules

A DELETE request is used to delete filtering rules from a DOTS server
(Figure 9).

```
DELETE /restconf/data/ietf-dots-access-control-list:access-lists\
       /client-identifier=dz6pHjaADkaFTbjr0JGBpw,iAYmCNPmrYoKoqzgFMiobw\
       /acl-name=sample-ipv4-acl&acl-type=ipv4-acl HTTP/1.1
Host: {host}:{port}
```

              Figure 9: DELETE to remove the filtering rules

If the DOTS server does not find the access list name and access list
type conveyed in the DELETE request in its configuration data, then
it responds with a 404 (Not Found) error response code. The DOTS
server successfully acknowledges a DOTS client's request to withdraw
the filtering rules using 204 (No Content) response code, and removes
the filtering rules as soon as possible.

3.3.3. Retrieving Installed Filtering Rules

   The DOTS client periodically queries the DOTS server to check the
   counters for installed filtering rules. A GET request is used to
   retrieve filtering rules from a DOTS server. Figure 10 shows how to
   retrieve all the filtering rules programmed by the DOTS client and
   the number of matches for the installed filtering rules.

```
 GET /restconf/data/ietf-dots-access-control-list:access-lists\
     /client-identifier=dz6pHjaADkaFTbjr0JGBpw,iAYmCNPmrYoKoqzgFMiobw?\
     content=all HTTP/1.1
 Host: {host}:{port}
 Accept: application/yang-data+json
```

       Figure 10: GET to retrieve the configuration data and state data for
                           the filtering rules

   If the DOTS server does not find the access list name and access list
   type conveyed in the GET request in its configuration data, then it
   responds with a 404 (Not Found) error response code.

4. IANA Considerations

   This specification registers new parameters for the DOTS data channel
   and establishes registries for mappings to JSON attributes.

4.1. DOTS Data Channel JSON Attribute Mappings Registry

   A new registry will be requested from IANA, entitled "DOTS data
   channel JSON attribute Mappings Registry". The registry is to be
   created as Expert Review Required.

4.2. Registration Template

   JSON Attribute:
      JSON attribute name.

   Description:
      Brief description of the attribute.

   Change Controller:
      For Standards Track RFCs, list the "IESG". For others, give the
      name of the responsible party. Other details (e.g., postal
      address, email address, home page URI) may also be included.

   Specification Document(s):
      Reference to the document or documents that specify the parameter,
      preferably including URIs that can be used to retrieve copies of

the documents. An indication of the relevant sections may also be
included but is not required.

4.3. Initial Registry Contents

   o JSON Attribute: "client-identifier"
   o Description: Client identifier.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "alias-name"
   o Description: Name of alias.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "traffic-protocol"
   o Description: Internet protocol numbers.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "port-range"
   o Description: The port range, lower-port for lower port number and
     upper-port for upper port number. For TCP, UDP, SCTP, or DCCP:
     the range of ports (e.g., 80 to 8080).
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "lower-port"
   o Description: Lower port number for port range.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "upper-port"
   o Description: Upper port number for port range.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "ip"
   o Description: IP address.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "prefix"
   o Description: IP prefix
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "fqdn"

o Description: Fully Qualified Domain Name, is the full name of a
   system, rather than just its hostname. For example, "venera" is a
   hostname, and "venera.isi.edu" is an FQDN.
o Change Controller: IESG
o Specification Document(s): this document

o JSON Attribute: "uri"
o Description: Uniform Resource Identifier (URI).
o Change Controller: IESG
o Specification Document(s): this document

5. Contributors

   The following individuals have contributed to this document:

   Dan Wing

   Email: dwing-ietf@fuggles.com

6. Security Considerations

   Authenticated encryption MUST be used for data confidentiality and
   message integrity. TLS based on client certificate MUST be used for
   mutual authentication. The interaction between the DOTS agents
   requires Transport Layer Security (TLS) with a cipher suite offering
   confidentiality protection and the guidance given in [RFC7525] MUST
   be followed to avoid attacks on TLS.

   An attacker may be able to inject RST packets, bogus application
   segments, etc., regardless of whether TLS authentication is used.
   Because the application data is TLS protected, this will not result
   in the application receiving bogus data, but it will constitute a DoS
   on the connection. This attack can be countered by using TCP-AO
   [RFC5925]. If TCP-AO is used, then any bogus packets injected by an
   attacker will be rejected by the TCP-AO integrity check and therefore
   will never reach the TLS layer.

   In order to prevent leaking internal information outside a client-
   domain, DOTS gateways located in the client-domain SHOULD NOT reveal
   the identity of internal DOTS clients (client-identifier) unless
   explicitly configured to do so.

   Special care should be taken in order to ensure that the activation
   of the proposed mechanism won't have an impact on the stability of
   the network (including connectivity and services delivered over that
   network).

   Involved functional elements in the cooperation system must establish
   exchange instructions and notification over a secure and
   authenticated channel. Adequate filters can be enforced to avoid
   that nodes outside a trusted domain can inject request such as
   deleting filtering rules. Nevertheless, attacks can be initiated
   from within the trusted domain if an entity has been corrupted.
   Adequate means to monitor trusted nodes should also be enabled.

7. Acknowledgements

   Thanks to Christian Jacquenet, Roland Dobbins, Roman Danyliw, Ehud
   Doron, Russ White, Jon Shallow, and Gilbert Clark for the discussion
   and comments.

8. References

8.1. Normative References

   [I-D.ietf-dots-architecture]
              Mortensen, A., Andreasen, F., Reddy, T.,
              christopher_gray3@cable.comcast.com, c., Compton, R., and
              N. Teague, "Distributed-Denial-of-Service Open Threat
              Signaling (DOTS) Architecture", draft-ietf-dots-
              architecture-05 (work in progress), October 2017.

   [I-D.ietf-dots-signal-channel]
              Reddy, T., Boucadair, M., Patil, P., Mortensen, A., and N.
              Teague, "Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-
              channel-06 (work in progress), October 2017.

   [I-D.ietf-netmod-acl-model]
              Jethanandani, M., Huang, L., Agarwal, S., and D. Blair,
              "Network Access Control List (ACL) YANG Data Model",
              draft-ietf-netmod-acl-model-14 (work in progress), October
              2017.

   [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>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
              Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
              June 2010, <https://www.rfc-editor.org/info/rfc5925>.

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

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
              RFC 7951, DOI 10.17487/RFC7951, August 2016,
              <https://www.rfc-editor.org/info/rfc7951>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <https://www.rfc-editor.org/info/rfc8040>.

8.2. Informative References

   [I-D.ietf-dots-requirements]
              Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed
              Denial of Service (DDoS) Open Threat Signaling
              Requirements", draft-ietf-dots-requirements-07 (work in
              progress), October 2017.

   [I-D.ietf-netmod-yang-tree-diagrams]
              Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-
              ietf-netmod-yang-tree-diagrams-02 (work in progress),
              October 2017.

   [IEEE.754.1985]
              Institute of Electrical and Electronics Engineers,
              "Standard for Binary Floating-Point Arithmetic", August
              1985.

   [proto_numbers]
              "IANA, "Protocol Numbers"", 2011,
              <http://www.iana.org/assignments/protocol-numbers>.

   [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <https://www.rfc-editor.org/info/rfc6020>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <https://www.rfc-editor.org/info/rfc6241>.

   [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport
              Layer Security (TLS) and Datagram Transport Layer Security
              (DTLS) Heartbeat Extension", RFC 6520,
              DOI 10.17487/RFC6520, February 2012,
              <https://www.rfc-editor.org/info/rfc6520>.

   [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
              Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March
              2014, <https://www.rfc-editor.org/info/rfc7159>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

Authors' Addresses

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

   Email: kondtir@gmail.com

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo   108-8118
Japan

Email: kaname@nttv6.jp

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu   210012
China

Email: frank.xialiang@huawei.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI   48104
United States

Email: amortensen@arbor.net

Nik Teague
Verisign, Inc.
United States

Email: nteague@verisign.com

Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel
                            Specification
                  draft-ietf-dots-data-channel-31

Abstract

   The document specifies a Distributed Denial-of-Service Open Threat
   Signaling (DOTS) data channel used for bulk exchange of data that
   cannot easily or appropriately communicated through the DOTS signal
   channel under attack conditions.

   This is a companion document to the DOTS signal channel
   specification.

Editorial Note (To be removed by RFC Editor)

   Please update these statements within the document with the RFC
   number to be assigned to this document:

   o "This version of this YANG module is part of RFC XXXX;"

   o "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling
      (DOTS) Data Channel Specification";

   o reference: RFC XXXX

   Please update the "revision" date of the YANG module.

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 23, 2020.

Table of Contents

1. Introduction

   A distributed denial-of-service (DDoS) attack is an attempt to make
   machines or network resources unavailable to their intended users.
   In most cases, sufficient scale can be achieved by compromising
   enough end-hosts and using those infected hosts to perpetrate and
   amplify the attack. The victim of such attack can be an application
   server, a router, a firewall, an entire network, etc.

   As discussed in [RFC8612], the lack of a common method to coordinate
   a real-time response among involved actors and network domains
   inhibits the speed and effectiveness of DDoS attack mitigation. From
   that standpoint, DDoS Open Threat Signaling (DOTS) defines an
   architecture that allows a DOTS client to send requests to a DOTS
   server for DDoS attack mitigation [I-D.ietf-dots-architecture]. The
   DOTS approach is thus meant to minimize the impact of DDoS attacks,
   thereby contributing to the enforcement of more efficient defensive
   if not proactive security strategies. To that aim, DOTS defines two
   channels: the signal and the data channels (Figure 1).

```
   +--------------+                                 +--------------+
   |              | <------- Signal Channel ------> |              |
   |  DOTS Client |                                 |  DOTS Server |
   |              | <======= Data Channel  ======> |              |
   +--------------+                                 +--------------+
```
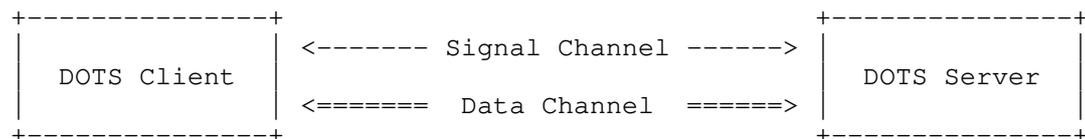
                          Figure 1: DOTS Channels

   The DOTS signal channel is used to carry information about a device
   or a network (or a part thereof) that is under a DDoS attack. Such
   information is sent by a DOTS client to an upstream DOTS server so
   that appropriate mitigation actions are undertaken on traffic deemed
   suspicious. The DOTS signal channel is further elaborated in
   [I-D.ietf-dots-signal-channel].

As for the DOTS data channel, it is used for infrequent bulk data
exchange between DOTS agents to significantly improve the
coordination of all the parties involved in the response to the
attack. Section 2 of [I-D.ietf-dots-architecture] mentions that the
DOTS data channel is used to perform the following tasks:

o Creating aliases for resources for which mitigation may be
   requested.

   A DOTS client may submit to its DOTS server a collection of
   prefixes which it would like to refer to by an alias when
   requesting mitigation. The DOTS server can respond to this
   request with either a success or failure response (see Section 2
   in [I-D.ietf-dots-architecture]).

   Refer to Section 6 for more details.

o Policy management, which enables a DOTS client to request the
   installation or withdrawal of traffic filters, dropping or rate-
   limiting unwanted traffic, and permitting accept-listed traffic.
   A DOTS client is entitled to instruct filtering rules only on IP
   resources that belong to its domain.

   Sample use cases for populating drop- or accept-list filtering
   rules are detailed hereafter:

   * If a network resource (DOTS client) is informed about a
      potential DDoS attack from a set of IP addresses, the DOTS
      client informs its servicing DOTS gateway of all suspect IP
      addresses that need to be drop-listed for further
      investigation. The DOTS client could also specify a list of
      protocols and port numbers in the drop-list rule.

      The DOTS gateway then propagates the drop-listed IP addresses
      to a DOTS server which will undertake appropriate actions so
      that traffic originated by these IP addresses to the target
      network (specified by the DOTS client) is blocked.

   * A network, that has partner sites from which only legitimate
      traffic arrives, may want to ensure that the traffic from these
      sites is not subjected to DDoS attack mitigation. The DOTS
      client uses the DOTS data channel to convey the accept-listed
      IP prefixes of the partner sites to its DOTS server.

      The DOTS server uses this information to accept-list flows
      originated by such IP prefixes and which reach the network.

   Refer to Section 7 for more details.

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
   [RFC2119] [RFC8174] when, and only when, they appear in all capitals,
   as shown here.

   The reader should be familiar with the terms defined in [RFC8612].

   The terminology for describing YANG modules is defined in [RFC7950].
   The meaning of the symbols in the tree diagrams is defined in
   [RFC8340].

   This document generalizes the notion of Access Control List (ACL) so
   that it is not device-specific [RFC8519]. As such, this document
   defines an ACL as an ordered set of rules that is used to filter
   traffic. Each rule is represented by an Access Control Entry (ACE).
   ACLs communicated via the DOTS data channel are not bound to a device
   interface.

   For the sake of simplicity, all of the examples in this document use
   "/restconf" as the discovered RESTCONF API root path. Many protocol
   header lines and message-body text within examples throughout the
   document are split into multiple lines for display purposes only.
   When a line ends with backslash ('\') as the last character, the line
   is wrapped for display purposes. It is to be considered to be joined
   to the next line by deleting the backslash, the following line break,
   and the leading whitespace of the next line.

3. DOTS Data Channel

3.1. Design Overview

   Unlike the DOTS signal channel, which must remain operational even
   when confronted with signal degradation due to packets loss, the DOTS
   data channel is not expected to be fully operational at all times,
   especially when a DDoS attack is underway. The requirements for a
   DOTS data channel protocol are documented in [RFC8612].

   This specification does not require an order of DOTS signal and data
   channel creations nor mandates a time interval between them. These
   considerations are implementation- and deployment-specific.

   As the primary function of the data channel is data exchange, a
   reliable transport mode is required in order for DOTS agents to
   detect data delivery success or failure. This document uses RESTCONF

[RFC8040] over TLS over TCP as the DOTS data channel protocol. The
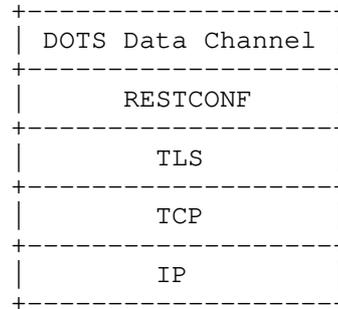abstract layering of DOTS data channel is shown in Figure 2.

```
+-------------------+
| DOTS Data Channel |
+-------------------+
|     RESTCONF      |
+-------------------+
|       TLS         |
+-------------------+
|       TCP         |
+-------------------+
|       IP          |
+-------------------+
```

Figure 2: Abstract Layering of DOTS Data Channel

The HTTP POST, PUT, PATCH, and DELETE methods are used to edit data
resources represented by DOTS data channel YANG modules. These basic
edit operations allow the DOTS data channel running configuration to
be altered by a DOTS client. Rules for generating and processing
RESTCONF methods are defined in Section 4 of [RFC8040].

DOTS data channel configuration information as well as state
information can be retrieved with the GET method. An HTTP status-
line is returned for each request to report success or failure for
RESTCONF operations (Section 5.4 of [RFC8040]). The "error-tag"
provides more information about encountered errors (Section 7 of
[RFC8040]).

DOTS clients perform the root resource discovery procedure discussed
in Section 3.1 of [RFC8040] to determine the root of the RESTCONF
API. After discovering the RESTCONF API root, a DOTS client uses
this value as the initial part of the path in the request URI in any
subsequent request to the DOTS server. The DOTS server may support
the retrieval of the YANG modules it supports (Section 3.7 in
[RFC8040]). For example, a DOTS client may use RESTCONF to retrieve
the vendor-specific YANG modules supported by its DOTS server.

JavaScript Object Notation (JSON) [RFC8259] payloads are used to
propagate the DOTS data-channel-specific payload messages that carry
request parameters and response information, such as errors. This
specification uses the encoding rules defined in [RFC7951] for
representing DOTS data channel configuration data using YANG
(Section 4) as JSON text.

A DOTS client registers itself to its DOTS server(s) in order to set
up DOTS data channel-related configuration data and receive state

data (i.e., non-configuration data) from the DOTS server(s)
(Section 5). Mutual authentication considerations are specified in
Section 8 of [I-D.ietf-dots-signal-channel]. The coupling of signal
and data channels is discussed in Section 4.4.1 of
[I-D.ietf-dots-signal-channel].

A DOTS client can either maintain a persistent connection or periodic
connections with its DOTS server(s). If the DOTS client needs to
frequently update the drop-list or accept-list filtering rules or
aliases, it maintains a persistent connection with the DOTS server.
For example, CAPTCHA and cryptographic puzzles can be used by the
mitigation service in the DOTS client domain to determine whether the
IP address is used for legitimate purpose or not, and the DOTS client
can frequently update the drop-list filtering rules. A persistent
connection is also useful if the DOTS client subscribes to event
notifications (Section 6.3 of [RFC8040]). Additional considerations
related to RESTCONF connection management (including, configuring the
connection type or the reconnect strategy) can be found in
[I-D.ietf-netconf-restconf-client-server].

A single DOTS data channel between DOTS agents can be used to
exchange multiple requests and multiple responses. To reduce DOTS
client and DOTS server workload, DOTS clients SHOULD re-use the same
TLS session. While the communication to the DOTS server is
quiescent, the DOTS client MAY probe the server to ensure it has
maintained cryptographic state. Such probes can also keep alive
firewall and/or NAT bindings. A TLS heartbeat [RFC6520] verifies
that the DOTS server still has TLS state by returning a TLS message.

A DOTS server may detect conflicting filtering requests from distinct
DOTS clients which belong to the same domain. For example, a DOTS
client could request to drop-list a prefix by specifying the source
prefix, while another DOTS client could request to accept-list that
same source prefix, but both having the same destination prefix.
DOTS servers SHOULD support a configuration parameter to indicate the
behavior to follow when a conflict is detected (e.g., reject all,
reject the new request, notify an administrator for validation).
Section 7.2 specifies a default behavior when no instruction is
supplied to a DOTS server.

How a DOTS client synchronizes its configuration with the one
maintained by its DOTS server(s) is implementation-specific. For
example:

o a DOTS client can systematically send a GET message before and/or
   after a configuration change request.

o a DOTS client can re-establish the disconnected DOTS session after
   an attack is mitigated and sends a GET message before a
   configuration change request.

NAT considerations for the DOTS data channel are similar to those
discussed in Section 3 of [I-D.ietf-dots-signal-channel].

How filtering rules that are instantiated on a DOTS server are
translated into network configurations actions is out of scope of
this specification.

Some of the fields introduced in Section 4 are also discussed in
Sections 5, 6, and 7. These sections are authoritative for these
fields.

3.2. DOTS Server(s) Discovery

This document assumes that DOTS clients are provisioned with a way to
know how to reach their DOTS server(s), which could occur by a
variety of means (e.g., local configuration, or dynamic means such as
DHCP [I-D.ietf-dots-server-discovery]). The specification of such
means are out of scope of this document.

Likewise, it is out of scope of this document to specify the behavior
to be followed by a DOTS client to send DOTS requests when multiple
DOTS servers are provisioned (e.g., contact all DOTS servers, select
one DOTS server among the list).

3.3. DOTS Gateways

When a server-domain DOTS gateway is involved in DOTS data channel
exchanges, the same considerations for manipulating the 'cdid'
(client domain identifier) parameter specified in
[I-D.ietf-dots-signal-channel] MUST be followed by DOTS agents. As a
reminder, 'cdid' is meant to assist the DOTS server to enforce some
policies (e.g., limit the number of filtering rules per DOTS client
or per DOTS client domain). A loop detect mechanism for DOTS
gateways is specified in Section 3.4.

If a DOTS gateway is involved, the DOTS gateway verifies that the
DOTS client is authorized to undertake a data channel action (e.g.,
instantiate filtering rules). If the DOTS client is authorized, it
propagates the rules to the upstream DOTS server. Likewise, the DOTS
server verifies that the DOTS gateway is authorized to relay data
channel actions. For example, to create or purge filters, a DOTS
client sends its request to its DOTS gateway. The DOTS gateway
validates the rules in the request and proxies the requests
containing the filtering rules to its DOTS server. When the DOTS

gateway receives the associated response from the DOTS server, it
propagates the response back to the DOTS client.

3.4. Detect and Prevent Infinite Loops

In order to detect and prevent infinite loops, DOTS gateways MUST
support the procedure defined in Section 5.7.1 of [RFC7230]. In
particular, each intermediate DOTS gateway MUST check that none of
its own information (e.g., server names, literal IP addresses) is
present in the "Via" header field of a DOTS message it receives:

o If it detects that its own information is present in the "Via"
   header field, the DOTS gateway MUST NOT forward the DOTS message.
   Messages that cannot be forwarded because of a loop SHOULD be
   logged with a "508 Loop Detected" status-line returned sent back
   to the DOTS peer. The structure of the reported error is depicted
   in Figure 3.


```
error-app-tag:  loop-detected
error-tag:      operation-failed
error-type:     transport, application
error-info:     <via-header> : A copy of the Via header field when
                the loop was detected.
Description:    An infinite loop has been detected when forwarding
                a requests via a proxy.
```

Figure 3: Loop Detected Error

It is RECOMMENDED that DOTS clients and gateways support methods
to alert administrators about loop errors so that appropriate
actions are undertaken.

o Otherwise, the DOTS agent MUST update or insert the "Via" header
   by appending its own information.

Unless configured otherwise, DOTS gateways at the boundaries of a
DOTS client domain SHOULD remove the previous "Via" header field
information after checking for a loop before forwarding. This
behavior is required for topology hiding purposes but can also serve
to minimize potential conflicts that may arise if overlapping
information is used in distinct DOTS domains (e.g., private IPv4
addresses, non globally unique aliases).

3.5. Stale Entries

   In order to avoid stale entries, a lifetime is associated with alias
   and filtering entries created by DOTS clients. Also, DOTS servers
   may track the inactivity timeout of DOTS clients to detect stale
   entries.

4. DOTS Data Channel YANG Module

4.1. Generic Tree Structure

   The DOTS data channel YANG module (ietf-dots-data-channel) provides a
   method for DOTS clients to manage aliases for resources for which
   mitigation may be requested. Such aliases may be used in subsequent
   DOTS signal channel exchanges to refer more efficiently to the
   resources under attack.

   Note that the full module's tree has been split across several
   figures to aid the exposition of the various sub-trees.

   The tree structure for the DOTS alias is depicted in Figure 4.

```
module: ietf-dots-data-channel
   +--rw dots-data
      +--rw dots-client* [cuid]
      |  +--rw cuid              string
      |  +--rw cdid?             string
      |  +--rw aliases
      |  |  +--rw alias* [name]
      |  |     +--rw name                 string
      |  |     +--rw target-prefix*       inet:ip-prefix
      |  |     +--rw target-port-range* [lower-port]
      |  |     |  +--rw lower-port    inet:port-number
      |  |     |  +--rw upper-port?   inet:port-number
      |  |     +--rw target-protocol*     uint8
      |  |     +--rw target-fqdn*         inet:domain-name
      |  |     +--rw target-uri*          inet:uri
      |  |     +--ro pending-lifetime?    int32
      |  +--rw acls
      |     ...
      +--ro capabilities
         ...
```

                     Figure 4: DOTS Alias Subtree

   Also, the 'ietf-dots-data-channel' module provides a method for DOTS
   clients to manage filtering rules. Examples of filtering management
   in a DOTS context include, but not limited to:

o Drop-list management, which enables a DOTS client to inform a DOTS
   server about sources from which traffic should be discarded.

o Accept-list management, which enables a DOTS client to inform a
   DOTS server about sources from which traffic should always be
   accepted.

o Policy management, which enables a DOTS client to request the
   installation or withdrawal of traffic filters, dropping or rate-
   limiting unwanted traffic and permitting accept-listed traffic.

The tree structure for the DOTS filtering entries is depicted in
Figure 5.

Investigations into the prospect of augmenting 'ietf-access-control-
list' to meet DOTS requirements concluded that such a design approach
did not support many of the DOTS requirements, e.g.,

o Retrieve a filtering entry (or all entries) created by a DOTS
   client.

o Delete a filtering entry that was instantiated by a DOTS client.

Accordingly, new DOTS filtering entries (i.e., Access Control List
(ACL)) are defined that mimic the structure specified in [RFC8519].
Concretely, DOTS agents are assumed to manipulate an ordered list of
ACLs; each ACL contains a separately ordered list of Access Control
Entries (ACEs). Each ACE has a group of match and a group of action
criteria.

Once all the ACE entries have been iterated though with no match,
then all the following ACL's ACE entries are iterated through until
the first match at which point the specified action is applied. If
there is no match during 'idle' time (i.e., no mitigation is active),
then there is no further action to be taken against the packet. If
there is no match during active mitigation, then the packet will
still be scrubbed by the DDoS mitigator.

```
module: ietf-dots-data-channel
   +--rw dots-data
      +--rw dots-client* [cuid]
      |  +--rw cuid              string
      |  +--rw cdid?             string
      |  +--rw aliases
      |  |  ...
      |  +--rw acls
      |     +--rw acl* [name]
      |        +--rw name                 string
      |        +--rw type?                ietf-acl:acl-type
      |        +--rw activation-type?     activation-type
      |        +--ro pending-lifetime?    int32
      |        +--rw aces
      |           +--rw ace* [name]
      |              +--rw name           string
      |              +--rw matches
      |              |  +--rw (l3)?
      |              |  |  +--:(ipv4)
      |              |  |  |  ...
      |              |  |  +--:(ipv6)
      |              |  |     ...
      |              |  +--rw (l4)?
      |              |     +--:(tcp)
      |              |     |  ...
      |              |     +--:(udp)
      |              |     |  ...
      |              |     +--:(icmp)
      |              |        ...
      |              +--rw actions
      |              |  +--rw forwarding    identityref
      |              |  +--rw rate-limit?   decimal64
      |              +--ro statistics
      |                 +--ro matched-packets?   yang:counter64
      |                 +--ro matched-octets?    yang:counter64
      +--ro capabilities
         ...
```

                        Figure 5: DOTS ACLs Subtree

   Filtering rules instructed by a DOTS client assumes a default
   direction: the destination is the DOTS client domain.

   DOTS forwarding actions can be 'accept' (i.e., accept matching
   traffic) or 'drop' (i.e., drop matching traffic without sending any
   ICMP error message). Accepted traffic can be subject to rate-
   limiting 'rate-limit'. Note that 'reject' action (i.e., drop
   matching traffic and send an ICMP error message to the source) is not

supported in 'ietf-dots-data-channel' because it is not appropriate
in the context of DDoS mitigation. Generating ICMP messages to
notify drops when mitigating a DDoS attack will exacerbate the DDoS
attack. Furthermore, these ICMP messages will be used by an attacker
as an explicit signal that the traffic is being blocked.

4.2. Filtering Fields

The 'ietf-dots-data-channel' module reuses the packet fields module
'ietf-packet-fields' [RFC8519] which defines matching on fields in
the packet including IPv4, IPv6, and transport layer fields. The
'ietf-dots-data-channel' module can be augmented, for example, to
support additional protocol-specific matching fields.

This specification defines a new IPv4/IPv6 matching field called
'fragment' to efficiently handle fragment-related filtering rules.
Indeed, [RFC8519] does not support such capability for IPv6 but
offers a partial support for IPv4 by means of 'flags'. Nevertheless,
the use of 'flags' is problematic since it does not allow to define a
bitmask. For example, setting other bits not covered by the 'flags'
filtering clause in a packet will allow that packet to get through
(because it won't match the ACE). Sample examples to illustrate how
'fragment' can be used are provided in Appendix A.

Figure 6 shows the IPv4 match subtree.

```
module: ietf-dots-data-channel
  +--rw dots-data
     +--rw dots-client* [cuid]
     |  ...
     |  +--rw acls
     |     +--rw acl* [name]
     |        ...
     |        +--rw aces
     |           +--rw ace* [name]
     |              +--rw name          string
     |              +--rw matches
     |              |  +--rw (l3)?
     |              |  |  +--:(ipv4)
     |              |  |  |  +--rw ipv4
     |              |  |  |     +--rw dscp?                  inet:dscp
     |              |  |  |     +--rw ecn?                   uint8
     |              |  |  |     +--rw length?                uint16
     |              |  |  |     +--rw ttl?                   uint8
     |              |  |  |     +--rw protocol?              uint8
     |              |  |  |     +--rw ihl?                   uint8
     |              |  |  |     +--rw flags?                 bits
     |              |  |  |     +--rw offset?                uint16
     |              |  |  |     +--rw identification?        uint16
     |              |  |  |     +--rw (destination-network)?
     |              |  |  |     |  +--:(destination-ipv4-network)
     |              |  |  |     |     +--rw destination-ipv4-network?
     |              |  |  |     |             inet:ipv4-prefix
     |              |  |  |     +--rw (source-network)?
     |              |  |  |     |  +--:(source-ipv4-network)
     |              |  |  |     |     +--rw source-ipv4-network?
     |              |  |  |     |             inet:ipv4-prefix
     |              |  |  |     +--rw fragment
     |              |  |  |        +--rw operator?       operator
     |              |  |  |        +--rw type        fragment-type
     |              |  |  +--:(ipv6)
     |              |  |  |  ...
     |              |  +--rw (l4)?
     |              |     ...
     |              +--rw actions
     |              |  ...
     |              +--ro statistics
     |                 ...
     +--ro capabilities
        ...
```

                 Figure 6: DOTS ACLs Subtree (IPv4 Match)

   Figure 7 shows the IPv6 match subtree.

```
module: ietf-dots-data-channel
  +--rw dots-data
     +--rw dots-client* [cuid]
     |  ...
     |  +--rw acls
     |     +--rw acl* [name]
     |        ...
     |        +--rw aces
     |           +--rw ace* [name]
     |              +--rw name           string
     |              +--rw matches
     |              |  +--rw (l3)?
     |              |  |  +--:(ipv4)
     |              |  |  |  ...
     |              |  |  +--:(ipv6)
     |              |  |     +--rw ipv6
     |              |  |        +--rw dscp?                  inet:dscp
     |              |  |        +--rw ecn?                   uint8
     |              |  |        +--rw length?                uint16
     |              |  |        +--rw ttl?                   uint8
     |              |  |        +--rw protocol?              uint8
     |              |  |        +--rw (destination-network)?
     |              |  |        |  +--:(destination-ipv6-network)
     |              |  |        |     +--rw destination-ipv6-network?
     |              |  |        |           inet:ipv6-prefix
     |              |  |        +--rw (source-network)?
     |              |  |        |  +--:(source-ipv6-network)
     |              |  |        |     +--rw source-ipv6-network?
     |              |  |        |           inet:ipv6-prefix
     |              |  |        +--rw flow-label?
     |              |  |        |     inet:ipv6-flow-label
     |              |  |        +--rw fragment
     |              |  |           +--rw operator?        operator
     |              |  |           +--rw type        fragment-type
     |              |  +--rw (l4)?
     |              |  |  ...
     |              +--rw actions
     |              |  ...
     |              +--ro statistics
     |                 ...
     +--ro capabilities
        ...
```

                Figure 7: DOTS ACLs Subtree (IPv6 Match)

   Figure 8 shows the TCP match subtree. In addition to the fields
   defined in [RFC8519], this specification defines a new TCP matching

field, called 'flags-bitmask', to efficiently handle TCP flags
filtering rules. Some examples are provided in Appendix B.

```
module: ietf-dots-data-channel
+--rw dots-data
   +-rw dots-client* [cuid]
    | ...
    | +-rw acls
    |   +-rw acl* [name]
    |     ...
    |      +-rw aces
    |         +-rw ace* [name]
    |            +-rw name          string
    |               +-rw matches
    |                | +-rw (l3)?
    |                | | ...
    |                | +-rw (l4)?
    |                |    +-:(tcp)
    |                |    | +-rw tcp
    |                |    |   +--rw sequence-number?          uint32
    |                |    |   +--rw acknowledgement-number?   uint32
    |                |    |   +--rw data-offset?              uint8
    |                |    |   +--rw reserved?                 uint8
    |                |    |   +--rw flags?                    bits
    |                |    |   +--rw window-size?              uint16
    |                |    |   +--rw urgent-pointer?           uint16
    |                |    |   +--rw options?                  binary
    |                |    |   +--rw flags-bitmask
    |                |    |   |  +--rw operator?              operator
    |                |    |   |  +--rw bitmask                uint16
    |                |    |   +--rw (source-port)?
    |                |    |   | +--:(source-port-range-or-operator)
    |                |    |   |    +--rw source-port-range-or-operator
    |                |    |   |       +--rw (port-range-or-operator)?
    |                |    |   |          +--:(range)
    |                |    |   |          | +--rw lower-port
    |                |    |   |          |       inet:port-number
    |                |    |   |          +--rw upper-port
    |                |    |   |                  inet:port-number
    |                |    |   |          +--:(operator)
    |                |    |   |             +--rw operator?
    |                |    |   |             |    operator
    |                |    |   |             +--rw port
    |                |    |   |                    inet:port-number
    |                |    |   +--rw (destination-port)?
    |                |    |      +--:(destination-port-range-or-operator)
    |                |    |         +--rw destination-port-range-or-operator
    |                |    |            +--rw (port-range-or-operator)?
```

```
         |        |  |                        +--:(range)
         |        |  |                        | +--rw lower-port
         |        |  |                        | |      inet:port-number
         |        |  |                        | +--rw upper-port
         |        |  |                        |        inet:port-number
         |        |  |                        +--:(operator)
         |        |  |                           +--rw operator?
         |        |  |                           |      operator
         |        |  |                           +--rw port
         |        |  |                                  inet:port-number
         |        |  +-:(udp)
         |        |  | ...
         |        |  +-:(icmp)
         |        |     ...
         |        +-rw actions
         |        | ...
         |        +-ro statistics
         |            ...
         +-ro capabilities
            ...
```
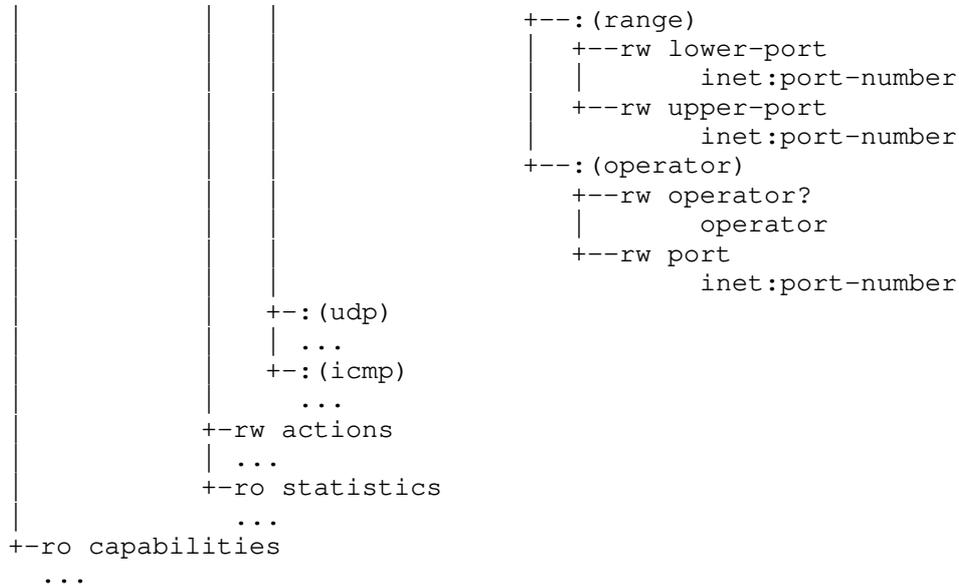
                   Figure 8: DOTS ACLs Subtree (TCP Match)

   Figure 9 shows the UDP and ICMP match subtrees. The same structure
   is used for both ICMP and ICMPv6. The indication whether an ACL is
   about ICMP or ICMPv6 is governed by the 'l3' match or the ACL type.

```
 module: ietf-dots-data-channel
  +-rw dots-data
    +-rw dots-client* [cuid]
    |  ...
    |  +-rw acls
    |    +-rw acl* [name]
    |      ...
    |      +-rw aces
    |        +-rw ace* [name]
    |          +--rw name          string
    |          +--rw matches
    |          |  +--rw (l3)?
    |          |  | ...
    |          |  +--rw (l4)?
    |          |     +--:(tcp)
    |          |     | ...
    |          |     +--:(udp)
    |          |     |  +--rw udp
    |          |     |     +--rw length?           uint16
    |          |     |     +--rw (source-port)?
    |          |     |     | +--:(source-port-range-or-operator)
```

```
|  |  |  |  |              +--rw source-port-range-or-operator
|  |  |  |  |                 +--rw (port-range-or-operator)?
|  |  |  |  |                    +--:(range)
|  |  |  |  |                    |  +--rw lower-port
|  |  |  |  |                    |  |      inet:port-number
|  |  |  |  |                    |  +--rw upper-port
|  |  |  |  |                    |         inet:port-number
|  |  |  |  |                    +--:(operator)
|  |  |  |  |                       +--rw operator?
|  |  |  |  |                       |      operator
|  |  |  |  |                       +--rw port
|  |  |  |  |                              inet:port-number
|  |  |  |  +--rw (destination-port)?
|  |  |  |     +--:(destination-port-range-or-operator)
|  |  |  |        +--rw destination-port-range-or-operator
|  |  |  |           +--rw (port-range-or-operator)?
|  |  |  |              +--:(range)
|  |  |  |              |  +--rw lower-port
|  |  |  |              |  |      inet:port-number
|  |  |  |              |  +--rw upper-port
|  |  |  |              |         inet:port-number
|  |  |  |              +--:(operator)
|  |  |  |                 +--rw operator?
|  |  |  |                 |      operator
|  |  |  |                 +--rw port
|  |  |  |                        inet:port-number
|  |  |  +--:(icmp)
|  |  |     +--rw icmp
|  |  |        +--rw type?            uint8
|  |  |        +--rw code?            uint8
|  |  |        +--rw rest-of-header?  binary
|  |  +--rw actions
|  |  |  ...
|  |  +--ro statistics
|  |     ...
+-ro capabilities
   ...
```

              Figure 9: DOTS ACLs Subtree (UDP and ICMP Match)

   DOTS implementations MUST support the following matching criteria:

      match based on the IP header (IPv4 and IPv6), match based on the
      transport header (TCP, UDP, and ICMP), and any combination
      thereof. The same matching fields are used for both ICMP and
      ICMPv6.

The following match fields MUST be supported by DOTS implementations
(Table 1):

```
ACL      Mandatory Fields
Match
-------  ------------------------------------------------------------
ipv4     length, protocol, destination-ipv4-network, source-
         ipv4-network, and fragment
ipv6     length, protocol, destination-ipv6-network, source-
         ipv6-network, and fragment
tcp      flags-bitmask, source-port-range-or-operator, and
         destination-port-range-or-operator
udp      length, source-port-range-or-operator, and destination-port-
         range-or-operator
icmp     type and code
```

                Table 1: Mandatory DOTS Channel Match Fields

Implementations MAY support other filtering match fields and actions.
The 'ietf-dots-data-channel' provides a method for an implementation
to expose its filtering capabilities. The tree structure of the
'capabilities' is shown in Figure 10. DOTS clients that support both
'fragment' and 'flags' (or 'flags-bitmask' and 'flags') matching
fields MUST NOT set these fields in the same request.

```
module: ietf-dots-data-channel
   +--rw dots-data
      ...
      +--ro capabilities
         +--ro address-family*       enumeration
         +--ro forwarding-actions*   identityref
         +--ro rate-limit?           boolean
         +--ro transport-protocols*  uint8
         +--ro ipv4
         |  +--ro dscp?                 boolean
         |  +--ro ecn?                  boolean
         |  +--ro length?               boolean
         |  +--ro ttl?                  boolean
         |  +--ro protocol?             boolean
         |  +--ro ihl?                  boolean
         |  +--ro flags?                boolean
         |  +--ro offset?               boolean
         |  +--ro identification?       boolean
         |  +--ro source-prefix?        boolean
         |  +--ro destination-prefix?   boolean
         |  +--ro fragment?             boolean
         +--ro ipv6
         |  +--ro dscp?                 boolean
```

```
            │  +--ro ecn?                   boolean
            │  +--ro length?                boolean
            │  +--ro hoplimit?              boolean
            │  +--ro protocol?              boolean
            │  +--ro destination-prefix?    boolean
            │  +--ro source-prefix?         boolean
            │  +--ro flow-label?            boolean
            │  +--ro fragment?              boolean
            +--ro tcp
            │  +--ro sequence-number?          boolean
            │  +--ro acknowledgement-number?   boolean
            │  +--ro data-offset?             boolean
            │  +--ro reserved?               boolean
            │  +--ro flags?                  boolean
            │  +--ro window-size?            boolean
            │  +--ro urgent-pointer?         boolean
            │  +--ro options?                boolean
            │  +--ro flags-bitmask?          boolean
            │  +--ro source-port?            boolean
            │  +--ro destination-port?       boolean
            │  +--ro port-range?             boolean
            +--ro udp
            │  +--ro length?             boolean
            │  +--ro source-port?        boolean
            │  +--ro destination-port?   boolean
            │  +--ro port-range?         boolean
            +--ro icmp
               +--ro type?            boolean
               +--ro code?            boolean
               +--ro rest-of-header?  boolean
```

               Figure 10: Filtering Capabilities Subtree

4.3. YANG Module

   This module uses the common YANG types defined in [RFC6991] and types
   defined in [RFC8519].

```
 <CODE BEGINS> file "ietf-dots-data-channel@2019-05-09.yang"
 module ietf-dots-data-channel {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-dots-data-channel";
   prefix data-channel;

   import ietf-inet-types {
     prefix inet;
     reference "Section 4 of RFC 6991";
   }
```

```
   import ietf-access-control-list {
     prefix ietf-acl;
     reference
       "RFC 8519: YANG Data Model for Network Access
                  Control Lists (ACLs)";
   }
   import ietf-packet-fields {
     prefix packet-fields;
     reference
       "RFC 8519: YANG Data Model for Network Access
                  Control Lists (ACLs)";
   }

   organization
     "IETF DDoS Open Threat Signaling (DOTS) Working Group";
   contact
     "WG Web:   <https://datatracker.ietf.org/wg/dots/>
      WG List:  <mailto:dots@ietf.org>

      Editor:   Mohamed Boucadair
                <mailto:mohamed.boucadair@orange.com>

      Editor:   Konda, Tirumaleswar Reddy
                <mailto:TirumaleswarReddy_Konda@McAfee.com>

      Author:   Jon Shallow
                <mailto:jon.shallow@nccgroup.com>

      Author:   Kaname Nishizuka
                <mailto:kaname@nttv6.jp>

      Author:   Liang Xia
                <mailto:frank.xialiang@huawei.com>

      Author:   Prashanth Patil
                <mailto:praspati@cisco.com>

      Author:   Andrew Mortensen
                <mailto:amortensen@arbor.net>

      Author:   Nik Teague
                <mailto:nteague@verisign.com>";
   description
     "This module contains YANG definition for configuring
      aliases for resources and filtering rules using DOTS
      data channel.

      Copyright (c) 2019 IETF Trust and the persons identified as
```

```
   revision 2019-05-09 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: Distributed Denial-of-Service Open Threat
                  Signaling (DOTS) Data Channel Specification";
   }

   typedef activation-type {
     type enumeration {
       enum "activate-when-mitigating" {
         value 1;
         description
           "The Access Control List (ACL) is installed only when
            a mitigation is active for the DOTS client.";
       }
       enum "immediate" {
         value 2;
         description
           "The ACL is immediately activated.";
       }
       enum "deactivate" {
         value 3;
         description
           "The ACL is maintained by the DOTS server, but it is
            deactivated.";
       }
     }
     description
       "Indicates the activation type of an ACL.";
   }

   typedef operator {
     type bits {
       bit not {
         position 0;
```

```
             description
               "If set, logical negation of operation.";
           }
         bit match {
           position 1;
           description
             "Match bit. This is a bitwise match operation
              defined as '(data & value) == value'.";
         }
         bit any {
           position 3;
           description
             "Any bit. This is a match on any of the bits in
              bitmask. It evaluates to 'true' if any of the bits
              in the value mask are set in the data,
              i.e., '(data & value) != 0'.";
         }
       }
       description
         "Specifies how to apply the defined bitmask.";
     }

   grouping tcp-flags {
     leaf operator {
       type operator;
       default "match";
       description
         "Specifies how to interpret the TCP flags.";
     }
     leaf bitmask {
       type uint16;
       mandatory true;
       description
         "The bitmask matches the last 4 bits of byte 12
          and byte 13 of the TCP header. For clarity, the 4 bits
          of byte 12 corresponding to the TCP data offset field
          are not included in any matching.";
     }
     description
       "Operations on TCP flags.";
   }

   typedef fragment-type {
     type bits {
       bit df {
         position 0;
         description
           "Don't fragment bit for IPv4.
```

```
            Must be set to 0 when it appears in an IPv6 filter.";
      }
      bit isf {
        position 1;
        description
          "Is a fragment.";
      }
      bit ff {
        position 2;
        description
          "First fragment.";
      }
      bit lf {
        position 3;
        description
          "Last fragment.";
      }
    }
    description
      "Different fragment types to match against.";
  }

  grouping target {
    description
      "Specifies the targets of the mitigation request.";
    leaf-list target-prefix {
      type inet:ip-prefix;
      description
        "IPv4 or IPv6 prefix identifying the target.";
    }
    list target-port-range {
      key "lower-port";
      description
        "Port range. When only lower-port is
         present, it represents a single port number.";
      leaf lower-port {
        type inet:port-number;
        mandatory true;
        description
          "Lower port number of the port range.";
      }
      leaf upper-port {
        type inet:port-number;
        must '. >= ../lower-port' {
          error-message
            "The upper port number must be greater than
             or equal to the lower-port number.";
          }
```

```
         description
           "Upper port number of the port range.";
       }
     }
     leaf-list target-protocol {
       type uint8;
       description
         "Identifies the target protocol number.

          Values are taken from the IANA protocol registry:
          https://www.iana.org/assignments/protocol-numbers/
          protocol-numbers.xhtml

          For example, 6 for TCP or 17 for UDP.";
     }
     leaf-list target-fqdn {
       type inet:domain-name;
       description
         "FQDN identifying the target.";
     }
     leaf-list target-uri {
       type inet:uri;
       description
         "URI identifying the target.";
     }
   }

   grouping fragment-fields {
     leaf operator {
       type operator;
       default "match";
       description
         "Specifies how to interpret the fragment type.";
     }
     leaf type {
       type fragment-type;
       mandatory true;
       description
         "Indicates what fragment type to look for.";
     }
     description
       "Operations on fragment types.";
   }

   grouping aliases {
     description
       "Top level container for aliases.";
     list alias {
```

```
          key "name";
          description
            "List of aliases.";
          leaf name {
            type string;
            description
              "The name of the alias.";
          }
          uses target;
          leaf pending-lifetime {
            type int32;
            units "minutes";
            config false;
            description
              "Indicates the pending validity lifetime of the alias
               entry.";
          }
        }
      }

    grouping ports {
      choice source-port {
        container source-port-range-or-operator {
          uses packet-fields:port-range-or-operator;
          description
            "Source port definition.";
        }
        description
          "Choice of specifying the source port or referring to
           a group of source port numbers.";
      }
      choice destination-port {
        container destination-port-range-or-operator {
          uses packet-fields:port-range-or-operator;
          description
            "Destination port definition.";
        }
        description
          "Choice of specifying a destination port or referring
           to a group of destination port numbers.";
      }
      description
        "Choice of specifying a source or destination port numbers.";
    }

    grouping access-lists {
      description
        "Specifies the ordered set of Access Control Lists.";
```

```
   list acl {
     key "name";
     ordered-by user;
     description
       "An ACL is an ordered list of Access Control Entries (ACE).
        Each ACE has a list of match criteria and a list of actions.";
     leaf name {
       type string {
         length "1..64";
       }
       description
         "The name of the access list.";
       reference
         "RFC 8519: YANG Data Model for Network Access
                    Control Lists (ACLs)";
     }
     leaf type {
       type ietf-acl:acl-type;
       description
         "Type of access control list. Indicates the primary intended
          type of match criteria (e.g., IPv4, IPv6) used in the list
          instance.";
       reference
         "RFC 8519: YANG Data Model for Network Access
                    Control Lists (ACLs)";
     }
     leaf activation-type {
       type activation-type;
       default "activate-when-mitigating";
       description
         "Indicates the activation type of an ACL. An ACL can be
          deactivated, installed immediately, or installed when
          a mitigation is active.";
     }
     leaf pending-lifetime {
       type int32;
       units "minutes";
       config false;
       description
         "Indicates the pending validity lifetime of the ACL
          entry.";
     }
     container aces {
       description
         "The Access Control Entries container contains
          a list of ACEs.";
       list ace {
         key "name";
```

```
            ordered-by user;
            description
              "List of access list entries.";
            leaf name {
              type string {
                length "1..64";
              }
              description
                "A unique name identifying this ACE.";
              reference
                "RFC 8519: YANG Data Model for Network Access
                          Control Lists (ACLs)";
            }
            container matches {
              description
                "The rules in this set determine what fields will be
                 matched upon before any action is taken on them.

                 If no matches are defined in a particular container,
                 then any packet will match that container.

                 If no matches are specified at all in an ACE, then any
                 packet will match the ACE.";
              reference
                "RFC 8519: YANG Data Model for Network Access
                          Control Lists (ACLs)";
              choice l3 {
                container ipv4 {
                  when "derived-from(../../../../type, "
                     + "'ietf-acl:ipv4-acl-type')";
                  uses packet-fields:acl-ip-header-fields;
                  uses packet-fields:acl-ipv4-header-fields;
                  container fragment {
                    description
                      "Indicates how to handle IPv4 fragments.";
                    uses fragment-fields;
                  }
                  description
                    "Rule set that matches IPv4 header.";
                }
                container ipv6 {
                  when "derived-from(../../../../type, "
                     + "'ietf-acl:ipv6-acl-type')";
                  uses packet-fields:acl-ip-header-fields;
                  uses packet-fields:acl-ipv6-header-fields;
                  container fragment {
                    description
                      "Indicates how to handle IPv6 fragments.";
```

```
                uses fragment-fields;
              }
              description
                "Rule set that matches IPv6 header.";
            }
            description
              "Either IPv4 or IPv6.";
          }
          choice l4 {
            container tcp {
              uses packet-fields:acl-tcp-header-fields;
              container flags-bitmask {
                description
                  "Indicates how to handle TCP flags.";
                uses tcp-flags;
              }
              uses ports;
              description
                "Rule set that matches TCP header.";
            }
            container udp {
              uses packet-fields:acl-udp-header-fields;
              uses ports;
              description
                "Rule set that matches UDP header.";
            }
            container icmp {
              uses packet-fields:acl-icmp-header-fields;
              description
                "Rule set that matches ICMP/ICMPv6 header.";
            }
            description
              "Can be TCP, UDP, or ICMP/ICMPv6";
          }
        }
        container actions {
          description
            "Definitions of action for this ACE.";
          leaf forwarding {
            type identityref {
              base ietf-acl:forwarding-action;
            }
            mandatory true;
            description
              "Specifies the forwarding action per ACE.";
            reference
              "RFC 8519: YANG Data Model for Network Access
                         Control Lists (ACLs)";
```

```
              }
              leaf rate-limit {
                when "../forwarding = 'ietf-acl:accept'" {
                  description
                    "Rate-limit is valid only when accept action is
                     used.";
                }
                type decimal64 {
                  fraction-digits 2;
                }
                units "bytes per second";
                description
                  "Specifies how to rate-limit the traffic.";
              }
            }
            container statistics {
              config false;
              description
                "Aggregate statistics.";
              uses ietf-acl:acl-counters;
            }
          }
        }
      }
    }

    container dots-data {
      description
        "Main container for DOTS data channel.";
      list dots-client {
        key "cuid";
        description
          "List of DOTS clients.";
        leaf cuid {
          type string;
          description
            "A unique identifier that is generated by a DOTS client
             to prevent request collisions.";
          reference
            "RFC YYYY: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel Specification";
        }
        leaf cdid {
          type string;
          description
            "A client domain identifier conveyed by a
             server-domain DOTS gateway to a remote DOTS server.";
          reference
```

```
          "RFC YYYY: Distributed Denial-of-Service Open Threat
                  Signaling (DOTS) Signal Channel Specification";
    }
    container aliases {
      description
        "Set of aliases that are bound to a DOTS client.";
      uses aliases;
    }
    container acls {
      description
        "Access lists that are bound to a DOTS client.";
      uses access-lists;
    }
  }
  container capabilities {
    config false;
    description
      "Match capabilities";
    leaf-list address-family {
      type enumeration {
        enum "ipv4" {
          description
            "IPv4 is supported.";
        }
        enum "ipv6" {
          description
            "IPv6 is supported.";
        }
      }
      description
        "Indicates the IP address families supported by
         the DOTS server.";
    }
    leaf-list forwarding-actions {
      type identityref {
        base ietf-acl:forwarding-action;
      }
      description
        "Supported forwarding action(s).";
    }
    leaf rate-limit {
      type boolean;
      description
        "Support of rate-limit action.";
    }
    leaf-list transport-protocols {
      type uint8;
      description
```

```
            "Upper-layer protocol associated with a filtering rule.

             Values are taken from the IANA protocol registry:
             https://www.iana.org/assignments/protocol-numbers/
             protocol-numbers.xhtml

             For example, this field contains 1 for ICMP, 6 for TCP
             17 for UDP, or 58 for ICMPv6.";
      }
      container ipv4 {
        description
          "Indicates IPv4 header fields that are supported to enforce
           ACLs.";
        leaf dscp {
          type boolean;
          description
            "Support of filtering based on Differentiated Services Code
             Point (DSCP).";
        }
        leaf ecn {
          type boolean;
          description
            "Support of filtering based on Explicit Congestion
             Notification (ECN).";
        }
        leaf length {
          type boolean;
          description
            "Support of filtering based on the Total Length.";
        }
        leaf ttl {
          type boolean;
          description
            "Support of filtering based on the Time to Live (TTL).";
        }
        leaf protocol {
          type boolean;
          description
            "Support of filtering based on protocol field.";
        }
        leaf ihl {
          type boolean;
          description
            "Support of filtering based on the Internet Header
             Length (IHL).";
        }
        leaf flags {
          type boolean;
```

```
        description
          "Support of filtering based on the 'flags'.";
      }
      leaf offset {
        type boolean;
        description
          "Support of filtering based on the 'offset'.";
      }
      leaf identification {
        type boolean;
        description
          "Support of filtering based on the 'identification'.";
      }
      leaf source-prefix {
        type boolean;
        description
          "Support of filtering based on the source prefix.";
      }
      leaf destination-prefix {
        type boolean;
        description
          "Support of filtering based on the destination prefix.";
      }
      leaf fragment {
        type boolean;
        description
          "Indicates the capability of a DOTS server to
           enforce filters on IPv4 fragments. That is, the match
           functionality based on the Layer 3 'fragment' clause
           is supported.";
      }
    }
    container ipv6 {
      description
        "Indicates IPv6 header fields that are supported to enforce
         ACLs.";
      leaf dscp {
        type boolean;
        description
          "Support of filtering based on DSCP.";
      }
      leaf ecn {
        type boolean;
        description
          "Support of filtering based on ECN.";
      }
      leaf length {
        type boolean;
```

```
          description
            "Support of filtering based on the Payload Length.";
        }
        leaf hoplimit {
          type boolean;
          description
            "Support of filtering based on the Hop Limit.";
        }
        leaf protocol {
          type boolean;
          description
            "Support of filtering based on the Next Header field.";
        }
        leaf destination-prefix {
          type boolean;
          description
            "Support of filtering based on the destination prefix.";
        }
        leaf source-prefix {
          type boolean;
          description
            "Support of filtering based on the source prefix.";
        }
        leaf flow-label {
          type boolean;
          description
            "Support of filtering based on the Flow label.";
        }
        leaf fragment {
          type boolean;
          description
            "Indicates the capability of a DOTS server to
             enforce filters on IPv6 fragments.";
        }
      }
      container tcp {
        description
          "Set of TCP fields that are supported by the DOTS server
           to enforce filters.";
        leaf sequence-number {
          type boolean;
          description
            "Support of filtering based on the TCP sequence number.";
        }
        leaf acknowledgement-number {
          type boolean;
          description
            "Support of filtering based on the TCP acknowledgement
```

```
                  number.";
            }
          leaf data-offset {
            type boolean;
            description
              "Support of filtering based on the TCP data-offset.";
          }
          leaf reserved {
            type boolean;
            description
              "Support of filtering based on the TCP reserved field.";
          }
          leaf flags {
            type boolean;
            description
              "Support of filtering, as defined in RFC 8519, based
               on the TCP flags.";
          }
          leaf window-size {
            type boolean;
            description
              "Support of filtering based on the TCP window size.";
          }
          leaf urgent-pointer {
            type boolean;
            description
              "Support of filtering based on the TCP urgent pointer.";
          }
          leaf options {
            type boolean;
            description
              "Support of filtering based on the TCP options.";
          }
          leaf flags-bitmask {
            type boolean;
            description
              "Support of filtering based on the TCP flags bitmask.";
          }
          leaf source-port {
            type boolean;
            description
              "Support of filtering based on the source port number.";
          }
          leaf destination-port {
            type boolean;
            description
              "Support of filtering based on the destination port
               number.";
```

```
          }
          leaf port-range {
            type boolean;
            description
              "Support of filtering based on a port range.

               This includes filtering based on a source port range,
               destination port range, or both. All operators
               (i.e, less than or equal to, greater than or equal to,
               equal to, and not equal to) are supported.";
          }
        }
        container udp {
          description
            "Set of UDP fields that are supported by the DOTS server
             to enforce filters.";
          leaf length {
            type boolean;
            description
              "Support of filtering based on the UDP length.";
          }
          leaf source-port {
            type boolean;
            description
              "Support of filtering based on the source port number.";
          }
          leaf destination-port {
            type boolean;
            description
              "Support of filtering based on the destination port
               number.";
          }
          leaf port-range {
            type boolean;
            description
              "Support of filtering based on a port range.

               This includes filtering based on a source port range,
               destination port range, or both. All operators
               (i.e, less than or equal, greater than or equal, equal to,
               and not equal to) are supported.";
          }
        }
        container icmp {
          description
            "Set of ICMP/ICMPv6 fields that are supported by the DOTS
             server to enforce filters.";
          leaf type {
```

```
              type boolean;
              description
                "Support of filtering based on the ICMP/ICMPv6 type.";
            }
            leaf code {
              type boolean;
              description
                "Support of filtering based on the ICMP/ICMPv6 code.";
            }
            leaf rest-of-header {
              type boolean;
              description
                "Support of filtering based on the ICMP four-bytes
                 field / the ICMPv6 message body.";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5. Managing DOTS Clients

5.1. Registering DOTS Clients

In order to make use of DOTS data channel, a DOTS client MUST
register to its DOTS server(s) by creating a DOTS client ('dots-
client') resource. To that aim, DOTS clients SHOULD send a POST
request (shown in Figure 11).

```
 POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
 Host: {host}:{port}
 Content-Type: application/yang-data+json


 {
   "ietf-dots-data-channel:dots-client": [
     {
       "cuid": "string"
     }
   ]
 }
```

                   Figure 11: POST to Register Schema

The 'cuid' (client unique identifier) parameter is described below:

cuid: A globally unique identifier that is meant to prevent
   collisions among DOTS clients. This attribute has the same

meaning, syntax, and processing rules as the 'cuid' attribute
defined in [I-D.ietf-dots-signal-channel].

DOTS clients MUST use the same 'cuid' for both signal and data
channels.

This is a mandatory attribute.

In deployments where server-domain DOTS gateways are enabled,
identity information about the origin source client domain SHOULD be
supplied to the DOTS server. That information is meant to assist the
DOTS server to enforce some policies. These policies can be enforced
per-client, per-client domain, or both. Figure 12 shows a schema
example of a request relayed by a server-domain DOTS gateway.

```
 POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
 Host: {host}:{port}
 Content-Type: application/yang-data+json

 {
   "ietf-dots-data-channel:dots-client": [
     {
       "cuid": "string",
       "cdid": "string"
     }
   ]
 }
```

Figure 12: POST to Register Schema (via a Server-Domain DOTS Gateway)

A server-domain DOTS gateway SHOULD add the following attribute:

cdid: This attribute has the same meaning, syntax, and processing
   rules as the 'cdid' attribute defined in
   [I-D.ietf-dots-signal-channel].

   In deployments where server-domain DOTS gateways are enabled,
   'cdid' does not need to be inserted when relaying DOTS methods to
   manage aliases (Section 6) or filtering rules (Section 7). DOTS
   servers are responsible for maintaining the association between
   'cdid' and 'cuid' for policy enforcement purposes.

   This is an optional attribute.

A request example to create a 'dots-client' resource is depicted in
Figure 13. This request is relayed by a server-domain DOTS gateway
as hinted by the presence of the 'cdid' attribute.

```
   POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
   Host: example.com
   Content-Type: application/yang-data+json

   {
     "ietf-dots-data-channel:dots-client": [
       {
         "cuid": "dz6pHjaADkaFTbjr0JGBpw",
         "cdid": "7eeaf349529eb55ed50113"
       }
     ]
   }
```

              Figure 13: POST to Register (DOTS gateway)

   As a reminder, DOTS gateways may rewrite the 'cuid' used by peer DOTS
   clients (Section 4.4.1 of [I-D.ietf-dots-signal-channel]).

   DOTS servers can identify the DOTS client domain using the 'cdid'
   parameter or using the client's DNS name specified in the Subject
   Alternative Name extension's dNSName type in the client certificate
   [RFC6125].

   DOTS servers MUST limit the number of 'dots-client' resources to be
   created by the same DOTS client to 1 per request. Requests with
   multiple 'dots-client' resources MUST be rejected by DOTS servers.
   To that aim, the DOTS server MUST rely on the same procedure to
   unambiguously identify a DOTS client as discussed in Section 4.4.1 of
   [I-D.ietf-dots-signal-channel].

   The DOTS server indicates the result of processing the POST request
   using status-line codes. Status codes in the range "2xx" codes are
   success, "4xx" codes are some sort of invalid requests and "5xx"
   codes are returned if the DOTS server has erred or is incapable of
   accepting the creation of the 'dots-client' resource. In particular,

   o "201 Created" status-line is returned in the response, if the DOTS
      server has accepted the request.

   o "400 Bad Request" status-line is returned by the DOTS server, if
      the request does not include a 'cuid' parameter. The error-tag
      "missing-attribute" is used in this case.

   o "409 Conflict" status-line is returned to the requesting DOTS
      client, if the data resource already exists. The error-tag
      "resource-denied" is used in this case.

Once a DOTS client registers itself to a DOTS server, it can
create/delete/retrieve aliases (Section 6) and filtering rules
(Section 7).

A DOTS client MAY use the PUT request (Section 4.5 in [RFC8040]) to
register a DOTS client within the DOTS server. An example is shown
in Figure 14.

```
        PUT /restconf/data/ietf-dots-data-channel:dots-data\
            /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
        Host: example.com
        Content-Type: application/yang-data+json

        {
          "ietf-dots-data-channel:dots-client": [
            {
              "cuid": "dz6pHjaADkaFTbjr0JGBpw"
            }
          ]
        }
```

Figure 14: PUT to Register

The DOTS gateway that inserted a 'cdid' in a PUT request MUST strip
the 'cdid' parameter in the corresponding response before forwarding
the response to the DOTS client.

5.2. Unregistering DOTS Clients

A DOTS client de-registers from its DOTS server(s) by deleting the
'cuid' resource(s). Resources bound to this DOTS client will be
deleted by the DOTS server. An example of a de-register request is
shown in Figure 15.

```
 DELETE /restconf/data/ietf-dots-data-channel:dots-data\
        /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
 Host: example.com
```

Figure 15: De-register a DOTS Client

6. Managing DOTS Aliases

The following sub-sections define means for a DOTS client to create
aliases (Section 6.1), retrieve one or a list of aliases
(Section 6.2), and delete an alias (Section 6.3).

6.1. Create Aliases

   A POST or PUT request is used by a DOTS client to create aliases, for
   resources for which a mitigation may be requested. Such aliases may
   be used in subsequent DOTS signal channel exchanges to refer more
   efficiently to the resources under attack.

   DOTS clients within the same domain can create different aliases for
   the same resource.

   The structure of POST requests used to create aliases is shown in
   Figure 16.

```
 POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=cuid HTTP/1.1
 Host: {host}:{port}
 Content-Type: application/yang-data+json

 {
  "ietf-dots-data-channel:aliases": {
    "alias": [
      {
        "name": "string",
        "target-prefix": [
          "string"
        ],
        "target-port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
        ],
        "target-protocol": [
          integer
        ],
        "target-fqdn": [
          "string"
        ],
        "target-uri": [
          "string"
        ]
      }
    ]
  }
}
```

            Figure 16: POST to Create Aliases (Request Schema)

The parameters are described below:

name: Name of the alias.

   This is a mandatory attribute.

target-prefix: Prefixes are separated by commas. Prefixes are
   represented using Classless Inter-domain Routing (CIDR) notation
   [RFC4632]. As a reminder, the prefix length must be less than or
   equal to 32 (resp. 128) for IPv4 (resp. IPv6).

   The prefix list MUST NOT include broadcast, loopback, or multicast
   addresses. These addresses are considered as invalid values. In
   addition, the DOTS server MUST validate that these prefixes are
   within the scope of the DOTS client domain. Other validation
   checks may be supported by DOTS servers.

   This is an optional attribute.

target-port-range: A range of port numbers.

   The port range is defined by two bounds, a lower port number
   (lower-port) and an upper port number (upper-port). The range is
   considered to include both the lower and upper bounds.

   When only 'lower-port' is present, it represents a single port
   number.

   For TCP, UDP, Stream Control Transmission Protocol (SCTP)
   [RFC4960], or Datagram Congestion Control Protocol (DCCP)
   [RFC4340], the range of port numbers can be, for example,
   1024-65535.

   This is an optional attribute.

target-protocol: A list of protocols. Values are taken from the
   IANA protocol registry [proto_numbers].

   If 'target-protocol' is not specified, then the request applies to
   any protocol.

   This is an optional attribute.

target-fqdn: A list of Fully Qualified Domain Names (FQDNs)
   identifying resources under attack [RFC8499].

   How a name is passed to an underlying name resolution library is
   implementation- and deployment-specific. Nevertheless, once the

name is resolved into one or multiple IP addresses, DOTS servers MUST apply the same validation checks as those for 'target-prefix'.

The use of FQDNs may be suboptimal because it does not guarantee that the DOTS server will resolve a name to the same IP addresses that the DOTS client does.

This is an optional attribute.

target-uri: A list of Uniform Resource Identifiers (URIs) [RFC3986].

The same validation checks used for 'target-fqdn' MUST be followed by DOTS servers to validate a target URI.

This is an optional attribute.

In POST or PUT requests, at least one of the 'target-prefix', 'target-fqdn', or 'target-uri' attributes MUST be present. DOTS agents can safely ignore Vendor-Specific parameters they don't understand.

If more than one 'target-*' scope types (e.g., 'target-prefix' and 'target-fqdn' or 'target-fqdn' and 'target-uri') are included in a POST or PUT request, the DOTS server binds all resulting IP addresses/prefixes to the same resource.

Figure 17 shows a POST request to create an alias called "https1" for HTTPS servers with IP addresses 2001:db8:6401::1 and 2001:db8:6401::2 listening on TCP port number 443.

```
POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: www.example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:aliases": {
    "alias": [
      {
        "name": "https1",
        "target-protocol": [
          6
        ],
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-port-range": [
          {
            "lower-port": 443
          }
        ]
      }
    ]
  }
}
```

                  Figure 17: Example of a POST to Create an Alias

   "201 Created" status-line MUST be returned in the response if the
   DOTS server has accepted the alias.

   "409 Conflict" status-line MUST be returned to the requesting DOTS
   client, if the request is conflicting with an existing alias name.
   The error-tag "resource-denied" is used in this case.

   If the request is missing a mandatory attribute or it contains an
   invalid or unknown parameter, "400 Bad Request" status-line MUST be
   returned by the DOTS server. The error-tag is set to "missing-
   attribute", "invalid-value", or "unknown-element" as a function of
   the encountered error.

   If the request is received via a server-domain DOTS gateway, but the
   DOTS server does not maintain a 'cdid' for this 'cuid' while a 'cdid'
   is expected to be supplied, the DOTS server MUST reply with "403
   Forbidden" status-line and the error-tag "access-denied". Upon
   receipt of this message, the DOTS client MUST register (Section 5).

A DOTS client uses the PUT request to modify the aliases in the DOTS
server. In particular, a DOTS client MUST update its alias entries
upon change of the prefix indicated in the 'target-prefix'.

A DOTS server MUST maintain an alias for at least 10080 minutes (1
week). If no refresh request is seen from the DOTS client, the DOTS
server removes expired entries.

6.2. Retrieve Installed Aliases

A GET request is used to retrieve one or all installed aliases by a
DOTS client from a DOTS server (Section 3.3.1 in [RFC8040]). If no
'name' is included in the request, this is an indication that the
request is about retrieving all aliases instantiated by the DOTS
client.

Figure 18 shows an example to retrieve all the aliases that were
instantiated by the requesting DOTS client. The "content" query
parameter and its permitted values are defined in Section 4.8.1 of
[RFC8040].

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /aliases?content=all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

            Figure 18: GET to Retrieve All Installed Aliases

Figure 19 shows an example of the response message body that includes
all the aliases that are maintained by the DOTS server for the DOTS
client identified by the 'cuid' parameter.

```
   {
     "ietf-dots-data-channel:aliases": {
       "alias": [
         {
           "name": "Server1",
           "target-protocol": [
             6
           ],
           "target-prefix": [
             "2001:db8:6401::1/128",
             "2001:db8:6401::2/128"
           ],
           "target-port-range": [
             {
               "lower-port": 443
             }
           ],
           "pending-lifetime": 3596
         },
         {
           "name": "Server2",
           "target-protocol": [
             6
           ],
           "target-prefix": [
             "2001:db8:6401::10/128",
             "2001:db8:6401::20/128"
           ],
           "target-port-range": [
             {
               "lower-port": 80
             }
           ],
           "pending-lifetime": 9869
         }
       ]
     }
   }
```

   Figure 19: An Example of Response Body Listing All Installed Aliases

   Figure 20 shows an example of a GET request to retrieve the alias
   "Server2" that was instantiated by the DOTS client.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /aliases/alias=Server2?content=all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

                   Figure 20: GET to Retrieve an Alias

   If an alias name ('name') is included in the request, but the DOTS
   server does not find that alias name for this DOTS client in its
   configuration data, it MUST respond with a "404 Not Found" status-
   line.

6.3. Delete Aliases

   A DELETE request is used to delete an alias maintained by a DOTS
   server.

   If the DOTS server does not find the alias name, conveyed in the
   DELETE request, in its configuration data for this DOTS client, it
   MUST respond with a "404 Not Found" status-line.

   The DOTS server successfully acknowledges a DOTS client's request to
   remove the alias using "204 No Content" status-line in the response.

   Figure 21 shows an example of a request to delete an alias.

```
DELETE /restconf/data/ietf-dots-data-channel:dots-data\
       /dots-client=dz6pHjaADkaFTbjr0JGBpw\
       /aliases/alias=Server1 HTTP/1.1
Host: example.com
```

                       Figure 21: Delete an Alias

7. Managing DOTS Filtering Rules

   The following sub-sections define means for a DOTS client to retrieve
   DOTS filtering capabilities (Section 7.1), create filtering rules
   (Section 7.2), retrieve active filtering rules (Section 7.3), and
   delete a filtering rule (Section 7.4).

7.1. Retrieve DOTS Filtering Capabilities

   A DOTS client MAY send a GET request to retrieve the filtering
   capabilities supported by a DOTS server. Figure 22 shows an example
   of such request.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /capabilities HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Figure 22: GET to Retrieve the Capabilities of a DOTS Server

A DOTS client which issued a GET request to retrieve the filtering capabilities supported by its DOTS server, SHOULD NOT request for filtering actions that are not supported by that DOTS server.

Figure 23 shows an example of a response body received from a DOTS server which supports:

o IPv4, IPv6, TCP, UDP, ICMP, and ICMPv6 mandatory match criteria listed in Section 4.2.

o 'accept', 'drop', and 'rate-limit' actions.

```
     {
      "ietf-dots-data-channel:capabilities": {
        "address-family": ["ipv4", "ipv6"],
        "forwarding-actions": ["drop", "accept"],
        "rate-limit": true,
        "transport-protocols": [1, 6, 17, 58],
        "ipv4": {
          "length": true,
          "protocol": true,
          "destination-prefix": true,
          "source-prefix": true,
          "fragment": true
        },
        "ipv6": {
          "length": true,
          "protocol": true,
          "destination-prefix": true,
          "source-prefix": true,
          "fragment": true
        },
        "tcp": {
          "flags-bitmask": true,
          "source-port": true,
          "destination-port": true,
          "port-range": true
        },
        "udp": {
          "length": true,
          "source-port": true,
          "destination-port": true,
          "port-range": true
        },
        "icmp": {
          "type": true,
          "code": true
        }
      }
    }
```

       Figure 23: Reply to a GET Request with Filtering Capabilities
                              (Message Body)

7.2. Install Filtering Rules

   A POST or PUT request is used by a DOTS client to communicate
   filtering rules to a DOTS server.

Figure 24 shows a POST request example to block traffic from
192.0.2.0/24 and destined to 198.51.100.0/24. Other examples are
discussed in Appendix A.

```
POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
 "ietf-dots-data-channel:acls": {
   "acl": [
     {
       "name": "sample-ipv4-acl",
       "type": "ipv4-acl-type",
       "activation-type": "activate-when-mitigating",
       "aces": {
         "ace": [
           {
             "name": "rule1",
             "matches": {
               "ipv4": {
                 "destination-ipv4-network": "198.51.100.0/24",
                 "source-ipv4-network": "192.0.2.0/24"
               }
             },
             "actions": {
               "forwarding": "drop"
             }
           }
         ]
       }
     }
   ]
 }
}
```

              Figure 24: POST to Install Filtering Rules

The meaning of these parameters is as follows:

name: The name of the access list.

   This is a mandatory attribute.

type: Indicates the primary intended type of match criteria (e.g.,
   IPv4, IPv6). It is set to 'ipv4-acl-type' in the example of
   Figure 24.

    This is an optional attribute.

   activation-type: Indicates whether an ACL has to be activated
      (immediately or during mitigation time) or instantiated without
      being activated (deactivated). Deactivated ACLs can be activated
      using a variety of means such as manual configuration on a DOTS
      server or using the DOTS data channel.

      If this attribute is not provided, the DOTS server MUST use
      'activate-when-mitigating' as default value.

      When a mitigation is in progress, the DOTS server MUST only
      activate 'activate-when-mitigating' filters that are bound to the
      DOTS client that triggered the mitigation.

      This is an optional attribute.

   matches: Define criteria used to identify a flow on which to apply
      the rule. It can be "l3" (IPv4, IPv6) or "l4" (TCP, UDP, ..).
      The detailed match parameters are specified in Section 4.

      In the example depicted in Figure 24, an IPv4 matching criteria is
      used.

      This is an optional attribute.

   destination-ipv4-network: The destination IPv4 prefix. DOTS servers
      MUST validate that these prefixes are within the scope of the DOTS
      client domain. Other validation checks may be supported by DOTS
      servers. If this attribute is not provided, the DOTS server
      enforces the ACL on any destination IP address that belong to the
      DOTS client domain.

      This is a mandatory attribute in requests with an 'activation-
      type' set to 'immediate'.

   source-ipv4-network: The source IPv4 prefix.

      This is an optional attribute.

   actions: Actions in the forwarding ACL category can be "drop" or
      "accept". The "accept" action is used to accept-list traffic.
      The "drop" action is used to drop-list traffic.

      Accepted traffic may be subject to "rate-limit"; the allowed
      traffic rate is represented in bytes per second. This unit is the
      same as the one used for "traffic-rate" in [RFC5575].

This is a mandatory attribute.

The DOTS server indicates the result of processing the POST request using the status-line. Concretely, "201 Created" status-line MUST be returned in the response if the DOTS server has accepted the filtering rules. If the request is missing a mandatory attribute or contains an invalid or unknown parameter (e.g., a match field not supported by the DOTS server), "400 Bad Request" status-line MUST be returned by the DOTS server in the response. The error-tag is set to "missing-attribute", "invalid-value", or "unknown-element" as a function of the encountered error.

If the request is received via a server-domain DOTS gateway, but the DOTS server does not maintain a 'cdid' for this 'cuid' while a 'cdid' is expected to be supplied, the DOTS server MUST reply with "403 Forbidden" status-line and the error-tag "access-denied". Upon receipt of this message, the DOTS client MUST register (Figure 11).

If the request is conflicting with an existing filtering installed by another DOTS client of the domain, absent any local policy, the DOTS server returns "409 Conflict" status-line to the requesting DOTS client. The error-tag "resource-denied" is used in this case.

The "insert" query parameter (Section 4.8.5 of [RFC8040]) MAY be used to specify how an access control entry is inserted within an ACL and how an ACL is inserted within an ACL set.

The DOTS client uses the PUT request to modify its filtering rules maintained by the DOTS server. In particular, a DOTS client MUST update its filtering entries upon change of the destination-prefix. How such change is detected is out of scope.

A DOTS server MUST maintain a filtering rule for at least 10080 minutes (1 week). If no refresh request is seen from the DOTS client, the DOTS server removes expired entries. Typically, a refresh request is a PUT request which echoes the content of a response to a GET request with all of the read-only parameters stripped out (e.g., pending-lifetime).

7.3. Retrieve Installed Filtering Rules

A DOTS client periodically queries its DOTS server to check the counters for installed filtering rules. A GET request is used to retrieve filtering rules from a DOTS server. In order to indicate which type of data is requested in a GET request, the DOTS client sets adequately the "content" query parameter.

If the DOTS server does not find the access list name conveyed in the
GET request in its configuration data for this DOTS client, it
responds with a "404 Not Found" status-line.

In order to illustrate the intended behavior, consider the example
depicted in Figure 25. In reference to this example, the DOTS client
requests the creation of an immediate ACL called "test-acl-ipv6-udp".

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=test-acl-ipv6-udp HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "test-acl-ipv6-udp",
        "type": "ipv6-acl-type",
        "activation-type": "immediate",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8:6401::2/127",
                  "source-ipv6-network": "2001:db8:1234::/96",
                  "protocol": 17,
                  "flow-label": 10000
                },
                "udp": {
                  "source-port": {
                    "operator": "lte",
                    "port": 80
                  },
                  "destination-port": {
                    "operator": "neq",
                    "port": 1010
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

        Figure 25: Example of a PUT Request to Create a Filtering

The peer DOTS server follows the procedure specified in Section 7.2
to process the request. We consider in the following that a positive
response is sent back to the requesting DOTS client to confirm that
the "test-acl-ipv6-udp" ACL is successfully installed by the DOTS
server.

The DOTS client can issue a GET request to retrieve all its filtering
rules and the number of matches for the installed filtering rules as
illustrated in Figure 26. The "content" query parameter is set to
'all'. The message body of the response to this GET request is shown
in Figure 27.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /acls?content=all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

          Figure 26: Retrieve the Configuration Data and State Data for the
                         Filtering Rules (GET Request)

```
    {
      "ietf-dots-data-channel:acls": {
        "acl": [
          {
            "name": "test-acl-ipv6-udp",
            "type": "ipv6-acl-type",
            "activation-type": "immediate",
            "pending-lifetime":9080,
            "aces": {
              "ace": [
                {
                  "name": "my-test-ace",
                  "matches": {
                    "ipv6": {
                      "destination-ipv6-network": "2001:db8:6401::2/127",
                      "source-ipv6-network": "2001:db8:1234::/96",
                      "protocol": 17,
                      "flow-label": 10000
                    },
                    "udp": {
                      "source-port": {
                        "operator": "lte",
                        "port": 80
                      },
                      "destination-port": {
                        "operator": "neq",
                        "port": 1010
                      }
                    }
                  },
                  "actions": {
                    "forwarding": "accept"
                  }
                }
              ]
            }
          }
        ]
      }
    }
```

       Figure 27: Retrieve the Configuration Data and State Data for the
                   Filtering Rules (Response Message Body)

   Also, a DOTS client can issue a GET request to retrieve only
   configuration data related to an ACL as shown in Figure 28. It does
   so by setting the "content" query parameter to 'config'.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=test-acl-ipv6-udp?content=config HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Figure 28: Retrieve the Configuration Data for a Filtering Rule (GET
                            Request)

A response to this GET request is shown in Figure 29.

```
     {
       "ietf-dots-data-channel:acls": {
         "acl": [
           {
             "name": "test-acl-ipv6-udp",
             "type": "ipv6-acl-type",
             "activation-type": "immediate",
             "aces": {
               "ace": [
                 {
                   "name": "my-test-ace",
                   "matches": {
                     "ipv6": {
                       "destination-ipv6-network": "2001:db8:6401::2/127",
                       "source-ipv6-network": "2001:db8:1234::/96",
                       "protocol": 17,
                       "flow-label": 10000
                     },
                     "udp": {
                       "source-port": {
                         "operator": "lte",
                         "port": 80
                       },
                       "destination-port": {
                         "operator": "neq",
                         "port": 1010
                       }
                     }
                   },
                   "actions": {
                     "forwarding": "accept"
                   }
                 }
               ]
             }
           }
         ]
       }
     }
```

         Figure 29: Retrieve the Configuration Data for a Filtering Rule
                          (Response Message Body)

   A DOTS client can also issue a GET request with a "content" query
   parameter set to 'non-config' to exclusively retrieve non-
   configuration data bound to a given ACL as shown in Figure 30. A
   response to this GET request is shown in Figure 31.

```
   GET /restconf/data/ietf-dots-data-channel:dots-data\
       /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
       /acl=test-acl-ipv6-udp?content=non-config HTTP/1.1
    Host: example.com
    Accept: application/yang-data+json
```

Figure 30: Retrieve the Non-Configuration Data for a Filtering Rule
                            (GET Request)

```
{
  "ietf-dots-data-channel:acls":  {
    "acl": [
      {
        "name": "test-acl-ipv6-udp",
        "pending-lifetime": 8000,
        "aces": {
          "ace": [
            {
              "name": "my-test-ace"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 31: Retrieve the Non-Configuration Data for a Filtering Rule
                        (Response Message Body)

7.4. Remove Filtering Rules

   A DELETE request is used by a DOTS client to delete filtering rules
   from a DOTS server.

   If the DOTS server does not find the access list name carried in the
   DELETE request in its configuration data for this DOTS client, it
   MUST respond with a "404 Not Found" status-line. The DOTS server
   successfully acknowledges a DOTS client's request to withdraw the
   filtering rules using "204 No Content" status-line, and removes the
   filtering rules accordingly.

   Figure 32 shows an example of a request to remove the IPv4 ACL
   "sample-ipv4-acl" created in Section 7.2.

```
  DELETE  /restconf/data/ietf-dots-data-channel:dots-data\
          /dots-client=dz6pHjaADkaFTbjr0JGBpw/acls\
          /acl=sample-ipv4-acl HTTP/1.1
   Host: example.com
```

                Figure 32: Remove a Filtering Rule (DELETE Request)

   Figure 33 shows an example of a response received from the DOTS
   server to confirm the deletion of "sample-ipv4-acl".

```
   HTTP/1.1 204 No Content
   Server: Apache
   Date: Fri, 27 Jul 2018 10:05:15 GMT
   Cache-Control: no-cache
   Content-Type: application/yang-data+json
   Content-Length: 0
   Connection: Keep-Alive
```

                  Figure 33: Remove a Filtering Rule (Response)

8. Operational Considerations

   The following operational considerations should be taken into
   account:

   o DOTS servers MUST NOT enable both DOTS data channel and direct
      configuration, to avoid race conditions and inconsistent
      configurations arising from simultaneous updates from multiple
      sources.

   o DOTS agents SHOULD enable the DOTS data channel to configure
      aliases and ACLs, and only use direct configuration as a stop-gap
      mechanism to test DOTS signal channel with aliases and ACLs.
      Further, direct configuration SHOULD only be used when the on-path
      DOTS agents are within the same domain.

   o If a DOTS server has enabled direct configuration, it can reject
      the DOTS data channel connection using hard ICMP error [RFC1122]
      or RST (Reset) bit in the TCP header or reject the RESTCONF
      request using an error response containing a "503 Service
      Unavailable" status-line.

9. IANA Considerations

   This document requests IANA to register the following URI in the "ns"
   subregistry within the "IETF XML Registry" [RFC3688]:

                 URI: urn:ietf:params:xml:ns:yang:ietf-dots-data-channel
                 Registrant Contact: The IESG.
                 XML: N/A; the requested URI is an XML namespace.

   This document requests IANA to register the following YANG module in
   the "YANG Module Names" subregistry [RFC7950] within the "YANG
   Parameters" registry.

                 Name: ietf-dots-data-channel
                 Namespace: urn:ietf:params:xml:ns:yang:ietf-dots-data-channel
                 Prefix: data-channel
                 Reference: RFC XXXX

   This module is not maintained by IANA.

10. Security Considerations

   RESTCONF security considerations are discussed in [RFC8040]. In
   particular, DOTS agents MUST follow the security recommendations in
   Sections 2 and 12 of [RFC8040]. Also, DOTS agents MUST support the
   mutual authentication TLS profile discussed in Sections 7.1 and 8 of
   [I-D.ietf-dots-signal-channel].

   Authenticated encryption MUST be used for data confidentiality and
   message integrity. The interaction between the DOTS agents requires
   Transport Layer Security (TLS) with a cipher suite offering
   confidentiality protection and the guidance given in [RFC7525] MUST
   be followed to avoid attacks on TLS.

   The installation of drop- or accept-list rules using RESTCONF over
   TLS reveals the attacker IP addresses and legitimate IP addresses
   only to the DOTS server trusted by the DOTS client. The secure
   communication channel between DOTS agents provides privacy and
   prevents a network eavesdropper from directly gaining access to the
   drop- and accept-listed IP addresses.

   An attacker may be able to inject RST packets, bogus application
   segments, etc., regardless of whether TLS authentication is used.
   Because the application data is TLS protected, this will not result
   in the application receiving bogus data, but it will constitute a DoS
   on the connection. This attack can be countered by using TCP-AO
   [RFC5925]. If TCP-AO is used, then any bogus packets injected by an
   attacker will be rejected by the TCP-AO integrity check and therefore
   will never reach the TLS layer.

   In order to prevent leaking internal information outside a client-
   domain, client-side DOTS gateways SHOULD NOT reveal the identity of

internal DOTS clients (e.g., source IP address, client's hostname)
unless explicitly configured to do so.

DOTS servers MUST verify that requesting DOTS clients are entitled to
enforce filtering rules on a given IP prefix. That is, only
filtering rules on IP resources that belong to the DOTS client domain
can be authorized by a DOTS server. The exact mechanism for the DOTS
servers to validate that the target prefixes are within the scope of
the DOTS client domain is deployment-specific.

Rate-limiting DOTS requests, including those with new 'cuid' values,
from the same DOTS client defends against DoS attacks that would
result from varying the 'cuid' to exhaust DOTS server resources.
Rate-limit policies SHOULD be enforced on DOTS gateways (if deployed)
and DOTS servers.

Applying resources quota per DOTS client and/or per DOTS client
domain (e.g., limit the number of aliases and filters to be installed
by DOTS clients) prevents DOTS server resources to be aggressively
used by some DOTS clients and ensures, therefore, DDoS mitigation
usage fairness. Additionally, DOTS servers may limit the number of
DOTS clients that can be enabled per domain.

When FQDNs are used as targets, the DOTS server MUST rely upon DNS
privacy enabling protocols (e.g., DNS over TLS [RFC7858] or DoH
[RFC8484]) to prevent eavesdroppers from possibly identifying the
target resources protected by the DDoS mitigation service, and means
to ensure the target FQDN resolution is authentic (e.g., DNSSEC
[RFC4034]).

The presence of DOTS gateways may lead to infinite forwarding loops,
which is undesirable. To prevent and detect such loops, a mechanism
is defined in Section 3.4.

All data nodes defined in the YANG module which can be created,
modified, and deleted (i.e., config true, which is the default) are
considered sensitive. Write operations applied to these data nodes
without proper protection can negatively affect network operations.
This module reuses YANG structures from [RFC8519], and the security
considerations for those nodes continue to apply for this usage.
Appropriate security measures are recommended to prevent illegitimate
users from invoking DOTS data channel primitives. Nevertheless, an
attacker who can access a DOTS client is technically capable of
launching various attacks, such as:

o Setting an arbitrarily low rate-limit, which may prevent
   legitimate traffic from being forwarded (rate-limit).

   o Setting an arbitrarily high rate-limit, which may lead to the
      forwarding of illegitimate DDoS traffic (rate-limit).

   o Communicating invalid aliases to the server (alias), which will
      cause the failure of associating both data and signal channels.

   o Setting invalid ACL entries, which may prevent legitimate traffic
      from being forwarded. Likewise, invalid ACL entries may lead to
      forward DDoS traffic.

11. Contributing Authors

   The following individuals co-authored this document:

      Kaname Nishizuka
      NTT Communications
      GranPark 16F 3-4-1 Shibaura, Minato-ku
      Tokyo 108-8118
      Japan

      Email: kaname@nttv6.jp

      Liang Xia
      Huawei
      101 Software Avenue, Yuhuatai District
      Nanjing, Jiangsu 210012
      China

      Email: frank.xialiang@huawei.com

      Prashanth Patil
      Cisco Systems, Inc.

      Email: praspati@cisco.com

      Andrew Mortensen
      Arbor Networks, Inc.
      2727 S. State St
      Ann Arbor, MI 48104
      United States

      Email: andrew.mortensen@netscout.com

      Nik Teague
      Iron Mountain Data Centers
      United Kingdom

      Email: nteague@ironmountain.co.uk

12. Contributors

   The following individuals have contributed to this document:

   o Dan Wing, Email: dwing-ietf@fuggles.com

   o Jon Shallow, NCC Group, Email: jon.shallow@nccgroup.com

13. Acknowledgements

   Thanks to Christian Jacquenet, Roland Dobbins, Roman Danyliw, Ehud
   Doron, Russ White, Gilbert Clark, Kathleen Moriarty, Nesredien
   Suleiman, Roni Even, and Brian Trammel for the discussion and
   comments.

   The authors would like to give special thanks to Kaname Nishizuka and
   Jon Shallow for their efforts in implementing the protocol and
   performing interop testing at IETF Hackathons.

   Many thanks to Ben Kaduk for the detailed AD review.

   Thanks to Martin Bjorklund for the guidance on RESTCONF.

   Thanks to Alexey Melnikov, Adam Roach, Suresh Krishnan, Mirja
   Kuehlewind, and Warren Kumari for the review.

14. References

14.1. Normative References

   [I-D.ietf-dots-signal-channel]
             K, R., Boucadair, M., Patil, P., Mortensen, A., and N.
             Teague, "Distributed Denial-of-Service Open Threat
             Signaling (DOTS) Signal Channel Specification", draft-
             ietf-dots-signal-channel-35 (work in progress), July 2019.

   [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>.

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
             DOI 10.17487/RFC3688, January 2004,
             <https://www.rfc-editor.org/info/rfc3688>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
             (CIDR): The Internet Address Assignment and Aggregation
             Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
             2006, <https://www.rfc-editor.org/info/rfc4632>.

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
             Verification of Domain-Based Application Service Identity
             within Internet Public Key Infrastructure Using X.509
             (PKIX) Certificates in the Context of Transport Layer
             Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
             2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
              RFC 6991, DOI 10.17487/RFC6991, July 2013,
              <https://www.rfc-editor.org/info/rfc6991>.

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

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

   [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
              RFC 7951, DOI 10.17487/RFC7951, August 2016,
              <https://www.rfc-editor.org/info/rfc7951>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <https://www.rfc-editor.org/info/rfc8040>.

   [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>.

   [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
              Interchange Format", STD 90, RFC 8259,
              DOI 10.17487/RFC8259, December 2017,
              <https://www.rfc-editor.org/info/rfc8259>.

   [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair,
              "YANG Data Model for Network Access Control Lists (ACLs)",
              RFC 8519, DOI 10.17487/RFC8519, March 2019,
              <https://www.rfc-editor.org/info/rfc8519>.

14.2. Informative References

   [I-D.ietf-dots-architecture]
              Mortensen, A., K, R., Andreasen, F., Teague, N., and R.
              Compton, "Distributed-Denial-of-Service Open Threat
              Signaling (DOTS) Architecture", draft-ietf-dots-
              architecture-14 (work in progress), May 2019.

   [I-D.ietf-dots-server-discovery]
             Boucadair, M. and R. K, "Distributed-Denial-of-Service
             Open Threat Signaling (DOTS) Server Discovery", draft-
             ietf-dots-server-discovery-04 (work in progress), June
             2019.

   [I-D.ietf-netconf-restconf-client-server]
             Watsen, K., "RESTCONF Client and Server Models", draft-
             ietf-netconf-restconf-client-server-14 (work in progress),
             July 2019.

   [proto_numbers]
             "IANA, "Protocol Numbers"", 2011,
             <http://www.iana.org/assignments/protocol-numbers>.

   [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
             Communication Layers", STD 3, RFC 1122,
             DOI 10.17487/RFC1122, October 1989,
             <https://www.rfc-editor.org/info/rfc1122>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
             Resource Identifier (URI): Generic Syntax", STD 66,
             RFC 3986, DOI 10.17487/RFC3986, January 2005,
             <https://www.rfc-editor.org/info/rfc3986>.

   [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>.

   [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
             Congestion Control Protocol (DCCP)", RFC 4340,
             DOI 10.17487/RFC4340, March 2006,
             <https://www.rfc-editor.org/info/rfc4340>.

   [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol",
             RFC 4960, DOI 10.17487/RFC4960, September 2007,
             <https://www.rfc-editor.org/info/rfc4960>.

   [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J.,
             and D. McPherson, "Dissemination of Flow Specification
             Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009,
             <https://www.rfc-editor.org/info/rfc5575>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
             Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
             June 2010, <https://www.rfc-editor.org/info/rfc5925>.

   [RFC6520]   Seggelmann, R., Tuexen, M., and M. Williams, "Transport
               Layer Security (TLS) and Datagram Transport Layer Security
               (DTLS) Heartbeat Extension", RFC 6520,
               DOI 10.17487/RFC6520, February 2012,
               <https://www.rfc-editor.org/info/rfc6520>.

   [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>.

   [RFC8340]   Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
               BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
               <https://www.rfc-editor.org/info/rfc8340>.

   [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>.

   [RFC8612]   Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
               Threat Signaling (DOTS) Requirements", RFC 8612,
               DOI 10.17487/RFC8612, May 2019,
               <https://www.rfc-editor.org/info/rfc8612>.

Appendix A. Sample Examples: Filtering Fragments

   This specification strongly recommends the use of "fragment" for
   handling fragments.

   Figure 34 shows the content of the POST request to be issued by a
   DOTS client to its DOTS server to allow the traffic destined to
   198.51.100.0/24 and UDP port number 53, but to drop all fragmented
   packets. The following ACEs are defined (in this order):

   o "drop-all-fragments" ACE: discards all fragments.

   o "allow-dns-packets" ACE: accepts DNS packets destined to
      198.51.100.0/24.

    POST /restconf/data/ietf-dots-data-channel:dots-data\
        /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
    Host: example.com
    Content-Type: application/yang-data+json

```
 {
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "dns-fragments",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "drop-all-fragments",
              "matches": {
                "ipv4": {
                  "fragment": {
                    "operator": "match",
                    "type": "isf"
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            }
          ]
          "ace": [
            {
              "name": "allow-dns-packets",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "198.51.100.0/24"
                }
                "udp": {
                  "destination-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
 }
```

                Figure 34: Filtering IPv4 Fragmented Packets

Figure 35 shows a POST request example issued by a DOTS client to its DOTS server to allow the traffic destined to 2001:db8::/32 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

o "drop-all-fragments" ACE: discards all fragments (including atomic fragments). That is, IPv6 packets which include a Fragment header (44) are dropped.

o "allow-dns-packets" ACE: accepts DNS packets destined to 2001:db8::/32.

```
 POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
 Host: example.com
 Content-Type: application/yang-data+json

 {
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "dns-fragments",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "drop-all-fragments",
              "matches": {
                "ipv6": {
                  "fragment": {
                    "operator": "match",
                    "type": "isf"
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            }
          ]
          "ace": [
            {
              "name": "allow-dns-packets",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8::/32"
                }
                "udp": {
```

```
                 "destination-port": {
                   "operator": "eq",
                   "port": 53
                 }
               }
             },
             "actions": {
               "forwarding": "accept"
             }
           }
         ]
       }
     }
   ]
 }
}
```

                Figure 35: Filtering IPv6 Fragmented Packets

Appendix B. Sample Examples: Filtering TCP Messages

   This section provides sample examples to illustrate TCP-specific
   filtering based on the flag bits. These examples should not be
   interpreted as recommended filtering behaviors under specific DDoS
   attacks.

B.1. Discard TCP Null Attack

   Figure 36 shows an example of a DOTS request sent by a DOTS client to
   install immediately a filter to discard incoming TCP messages having
   all flags unset. The bitmask can be set to 255 to check against the
   (CWR, ECE, URG, ACK, PSH, RST, SYN, FIN) flags.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=tcp-flags-example HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [{
      "name": "tcp-flags-example",
      "activation-type": "immediate",
      "aces": {
        "ace": [{
          "name": "null-attack",
          "matches": {
            "tcp": {
              "flags-bitmask": {
                "operator": "not any",
                "bitmask": 4095
              }
            }
          },
          "actions": {
            "forwarding": "drop"
          }
        }]
      }
    }]
  }
}
```

Figure 36: Example of a DOTS Request to Deny TCP Null Attack Messages

B.2. Rate-Limit SYN Flooding

   Figure 37 shows an ACL example to rate-limit incoming SYNs during a
   SYN-flood attack.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=tcp-flags-example HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [{
      "name": "tcp-flags-example",
      "activation-type": "activate-when-mitigating",
      "aces": {
        "ace": [{
          "name": "rate-limit-syn",
          "matches": {
            "tcp": {
              "flags-bitmask": {
                "operator": "match",
                "bitmask": 2
              }
            }
          },
          "actions": {
            "forwarding": "accept",
            "rate-limit": "20.00"
          }
        }]
      }
    }]
  }
}
```

   Figure 37: Example of DOTS Request to Rate-Limit Incoming TCP SYNs

B.3. Rate-Limit ACK Flooding

   Figure 38 shows an ACL example to rate-limit incoming ACKs during an
   ACK-flood attack.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=tcp-flags-example HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [{
      "name": "tcp-flags-example",
      "type": "ipv4-acl-type",
      "activation-type": "activate-when-mitigating",
      "aces": {
        "ace": [{
          "name": "rate-limit-ack",
          "matches": {
            "tcp": {
              "flags-bitmask": {
                "operator": "match",
                "bitmask": 16
              }
            }
          },
          "actions": {
            "forwarding": "accept",
            "rate-limit": "20.00"
          }
        }]
      }
    }]
  }
}
```

   Figure 38: Example of DOTS Request to Rate-Limit Incoming TCP ACKs

Authors' Addresses

   Mohamed Boucadair (editor)
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

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

Email: kondtir@gmail.com

Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
                    draft-ietf-dots-requirements-07

Abstract

   This document defines the requirements for the Distributed Denial of
   Service (DDoS) Open Threat Signaling (DOTS) protocols coordinating
   attack response against DDoS attacks.

Table of Contents

1. Introduction

1.1. Context and Motivation

   Distributed Denial of Service (DDoS) attacks continue to plague
   network operators around the globe, from Tier-1 service providers on
   down to enterprises and small businesses. Attack scale and frequency
   similarly have continued to increase, in part as a result of software
   vulnerabilities leading to reflection and amplification attacks.
   Once-staggering attack traffic volume is now the norm, and the impact
   of larger-scale attacks attract the attention of international press
   agencies.

   The greater impact of contemporary DDoS attacks has led to increased
   focus on coordinated attack response. Many institutions and
   enterprises lack the resources or expertise to operate on-premises

attack mitigation solutions themselves, or simply find themselves
constrained by local bandwidth limitations. To address such gaps,
security service providers have begun to offer on-demand traffic
scrubbing services, which aim to separate the DDoS traffic from
legitimate traffic and forward only the latter. Today each such
service offers a proprietary invocation interface for subscribers to
request attack mitigation, tying subscribers to proprietary signaling
implementations while also limiting the subset of network elements
capable of participating in the attack mitigation. As a result of
signaling interface incompatibility, attack responses may be
fragmentary or otherwise incomplete, leaving key players in the
attack path unable to assist in the defense.

The lack of a common method to coordinate a real-time response among
involved actors and network domains inhibits the speed and
effectiveness of DDoS attack mitigation. This document describes the
required characteristics of protocols enabling requests for DDoS
attack mitigation, reducing attack impact and leading to more
efficient defensive strategies.

DDoS Open Threat Signaling (DOTS) communicates the need for defensive
action in anticipation of or in response to an attack, but does not
dictate the form any defensive action takes. DOTS supplements calls
for help with pertinent details about the detected attack, allowing
entities participating in DOTS to form ad hoc, adaptive alliances
against DDoS attacks as described in the DOTS use cases
[I-D.ietf-dots-use-cases]. The requirements in this document are
derived from those use cases and [I-D.ietf-dots-architecture].

1.2. Terminology

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

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic
   originating from multiple sources are directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability of servers, services, applications, and/or other
   functionality of an attack target. Denial-of-service
   considerations are discussed in detail in [RFC4732].

DDoS attack target: A network connected entity with a finite set of
   resources, such as network bandwidth, memory or CPU, that is the
   focus of a DDoS attack. Potential targets include (but not

limited to) network elements, network links, servers, and
services.

DDoS attack telemetry: Collected measurements and behavioral
characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions taken to recognize and
filter out DDoS attack traffic while passing legitimate traffic to
the attack target.

Mitigation: A set of countermeasures enforced against traffic
destined for the target or targets of a detected or reported DDoS
attack, where countermeasure enforcement is managed by an entity
in the network path between attack sources and the attack target.
Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of
performing mitigation of a detected or reported DDoS attack. For
the purposes of this document, this entity is a black box capable
of mitigation, making no assumptions about availability or design
of countermeasures, nor about the programmable interface(s)
between this entity and other network elements. The mitigator and
invoked DOTS server are assumed to belong to the same
administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting
attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to
messages from DOTS clients. The DOTS server enables mitigation on
behalf of the DOTS client, if requested, by communicating the DOTS
client's request to the mitigator and returning selected mitigator
feedback to the requesting DOTS client. A DOTS server may also be
colocated with a mitigator.

DOTS agent: Any DOTS-aware software module capable of participating
in a DOTS signal or data channel. It can be a DOTS client, DOTS
server, or, as a logical agent, a DOTS gateway.

DOTS gateway: A DOTS-aware software module resulting from the
logical concatenation of a DOTS server and a DOTS client,
analogous to a Session Initiation Protocol (SIP) [RFC3261] Back-
to-Back User Agent (B2BUA) [RFC7092]. Client-side DOTS gateways
are DOTS gateways that are in the DOTS client's domain, while
server-side DOTS gateways denote DOTS gateways that are in the
DOTS server's domain. DOTS gateways are discussed in detail in
[I-D.ietf-dots-architecture].

   Signal channel: A bidirectional, mutually authenticated
      communication channel between two DOTS agents characterized by
      resilience even in conditions leading to severe packet loss, such
      as a volumetric DDoS attack causing network congestion.

   DOTS signal: A concise authenticated status/control message
      transmitted between DOTS agents, used to indicate client's need
      for mitigation, as well as to convey the status of any requested
      mitigation.

   Heartbeat: A message transmitted between DOTS agents over the signal
      channel, used as a keep-alive and to measure peer health.

   Data channel: A secure communication layer between two DOTS agents
      used for infrequent bulk exchange of data not easily or
      appropriately communicated through the signal channel under attack
      conditions.

   Filter: A specification of a matching network traffic flow or set of
      flows. The filter will typically have a policy associated with
      it, e.g., rate-limiting or discarding matching traffic.

   Blacklist: A filter list of addresses, prefixes, and/or other
      identifiers indicating sources from which traffic should be
      blocked, regardless of traffic content.

   Whitelist: A list of addresses, prefixes, and/or other identifiers
      indicating sources from which traffic should always be allowed,
      regardless of contradictory data gleaned in a detected attack.

   Multi-homed DOTS client: A DOTS client exchanging messages with
      multiple DOTS servers, each in a separate administrative domain.

2. Requirements

   This section describes the required features and characteristics of
   the DOTS protocol.

   DOTS is an advisory protocol. An active DDoS attack against the
   entity controlling the DOTS client need not be present before
   establishing a communication channel between DOTS agents. Indeed,
   establishing a relationship with peer DOTS agents during normal
   network conditions provides the foundation for more rapid attack
   response against future attacks, as all interactions setting up DOTS,
   including any business or service level agreements, are already
   complete. Peer DOTS agents are provisioned to a DOTS client using a
   variety of manual or dynamic methods.

The DOTS protocol must at a minimum make it possible for a DOTS
client to request a mitigator's aid mounting a defense, coordinated
by a DOTS server, against a suspected attack, signaling within or
between domains as requested by local operators. DOTS clients should
similarly be able to withdraw aid requests. DOTS requires no
justification from DOTS clients for requests for help, nor do DOTS
clients need to justify withdrawing help requests: the decision is
local to the DOTS clients' domain. Multi-homed DOTS clients must be
able to select the appropriate DOTS server(s) to which a mitigation
request is to be sent. Further multi-homing considerations are out
of scope.

Regular feedback between DOTS clients and DOTS servers supplement the
defensive alliance by maintaining a common understanding of the DOTS
agents' health and activity. Bidirectional communication between
DOTS clients and DOTS servers is therefore critical.

DOTS protocol implementations face competing operational goals when
maintaining this bidirectional communication stream. On the one
hand, the protocol must be resilient under extremely hostile network
conditions, providing continued contact between DOTS agents even as
attack traffic saturates the link. Such resiliency may be developed
several ways, but characteristics such as small message size,
asynchronous, redundant message delivery and minimal connection
overhead (when possible given local network policy) will tend to
contribute to the robustness demanded by a viable DOTS protocol.
Operators of peer DOTS-enabled domains may enable quality- or class-
of-service traffic tagging to increase the probability of successful
DOTS signal delivery, but DOTS does not require such policies be in
place. The DOTS solution indeed must be viable especially in their
absence.

On the other hand, DOTS must include protections ensuring message
confidentiality, integrity and authenticity to keep the protocol from
becoming another vector for the very attacks it's meant to help fight
off. DOTS clients must be able to authenticate DOTS servers, and
vice versa, to avoid exposing new attack surfaces when deploying
DOTS; specifically, to prevent DDoS mitigation in response to DOTS
signaling from becoming a new form of attack. In order to provide
this level of protection, DOTS agents must have a way to negotiate
and agree upon the terms of protocol security. Attacks against the
transport protocol should not offer a means of attack against the
message confidentiality, integrity and authenticity.

The DOTS server and client must also have some common method of
defining the scope of any mitigation performed by the mitigator, as
well as making adjustments to other commonly configurable features,

such as listen port numbers, exchanging black- and white-lists, and
so on.

Finally, DOTS should be sufficiently extensible to meet future needs
in coordinated attack defense, although this consideration is
necessarily superseded by the other operational requirements.

2.1. General Requirements

GEN-001 Extensibility: Protocols and data models developed as part
   of DOTS MUST be extensible in order to keep DOTS adaptable to
   operational and proprietary DDoS defenses. Future extensions MUST
   be backward compatible. DOTS protocols MUST use a version number
   system to distinguish protocol revisions. Implementations of
   older protocol versions SHOULD ignore information added to DOTS
   messages as part of newer protocol versions.

GEN-002 Resilience and Robustness: The signaling protocol MUST be
   designed to maximize the probability of signal delivery even under
   the severely constrained network conditions imposed by particular
   attack traffic. The protocol MUST be resilient, that is, continue
   operating despite message loss and out-of-order or redundant
   message delivery. In support of signaling protocol robustness,
   DOTS signals SHOULD be conveyed over a transport not susceptible
   to Head of Line Blocking.

GEN-003 Bidirectionality: To support peer health detection, to
   maintain an open signal channel, and to increase the probability
   of signal delivery during attack, the signal channel MUST be
   bidirectional, with client and server transmitting signals to each
   other at regular intervals, regardless of any client request for
   mitigation. Unidirectional messages MUST be supported within the
   bidirectional signal channel to allow for unsolicited message
   delivery, enabling asynchronous notifications between agents.

GEN-004 Bulk Data Exchange: Infrequent bulk data exchange between
   DOTS agents can also significantly augment attack response
   coordination, permitting such tasks as population of black- or
   white-listed source addresses; address or prefix group aliasing;
   exchange of incident reports; and other hinting or configuration
   supplementing attack response.

   As the resilience requirements for the DOTS signal channel mandate
   small signal message size, a separate, secure data channel
   utilizing a reliable transport protocol MUST be used for bulk data
   exchange.

2.2. Signal Channel Requirements

   SIG-001 Use of Common Transport Protocols: DOTS MUST operate over
      common widely deployed and standardized transport protocols.
      While connectionless transport such as the User Datagram Protocol
      (UDP) [RFC0768] SHOULD be used for the signal channel, the
      Transmission Control Protocol (TCP) [RFC0793] MAY be used if
      necessary due to network policy or middlebox capabilities or
      configurations.

   SIG-002 Sub-MTU Message Size: To avoid message fragmentation and the
      consequently decreased probability of message delivery over a
      congested link, signaling protocol message size MUST be kept under
      signaling Path Maximum Transmission Unit (PMTU), including the
      byte overhead of any encapsulation, transport headers, and
      transport- or message-level security.

      DOTS agents SHOULD attempt to learn the PMTU through mechanisms
      such as Path MTU Discovery [RFC1191] or Packetization Layer Path
      MTU Discovery [RFC4821]. If the PMTU cannot be discovered, DOTS
      agents SHOULD assume a PMTU of 1280 bytes. If IPv4 support on
      legacy or otherwise unusual networks is a consideration and PMTU
      is unknown, DOTS implementations MAY rely on a PMTU of 576 bytes,
      as discussed in [RFC0791] and [RFC1122].

   SIG-003 Channel Health Monitoring: DOTS agents MUST support exchange
      of heartbeat messages over the signal channel to monitor channel
      health. Peer DOTS agents SHOULD regularly send heartbeats to each
      other while a mitigation request is active. The heartbeat
      interval during active mitigation is not specified, but SHOULD be
      frequent enough to maintain any on-path NAT bindings during
      mitigation.

      To support scenarios in which loss of heartbeat is used to trigger
      mitigation, and to keep the channel active, DOTS clients MAY
      solicit heartbeat exchanges after successful mutual
      authentication. When DOTS agents are exchanging heartbeats and no
      mitigation request is active, either agent MAY request changes to
      the heartbeat rate. For example, a DOTS server might want to
      delay or cease heartbeat exchanges when an active DOTS client has
      not requested mitigation, in order to control load.

      Following mutual authentication, a signal channel MUST be
      considered active until a DOTS agent explicitly ends the session,
      or either DOTS agent fails to receive heartbeats from the other
      after a mutually agreed upon timeout period has elapsed. Because
      heartbeat loss is much more likely during volumetric attack, DOTS
      agents SHOULD avoid signal channel termination when mitigation is

active and heartbeats are not received by either DOTS agent for an
extended period. In such circumstances, DOTS clients MAY attempt
to reestablish the signal channel. DOTS servers SHOULD monitor
the attack, using feedback from the mitigator and other available
sources, and MAY use the absence of attack traffic and lack of
client heartbeats as an indication the signal channel is defunct.

SIG-004 Channel Redirection: In order to increase DOTS operational
flexibility and scalability, DOTS servers SHOULD be able to
redirect DOTS clients to another DOTS server at any time. DOTS
clients MUST NOT assume the redirection target DOTS server shares
security state with the redirecting DOTS server. DOTS clients MAY
attempt abbreviated security negotiation methods supported by the
protocol, such as DTLS session resumption, but MUST be prepared to
negotiate new security state with the redirection target DOTS
server.

Due to the increased likelihood of packet loss caused by link
congestion during an attack, DOTS servers SHOULD NOT redirect
while mitigation is enabled during an active attack against a
target in the DOTS client's domain.

SIG-005 Mitigation Requests and Status: Authorized DOTS clients MUST
be able to request scoped mitigation from DOTS servers. DOTS
servers MUST send mitigation request status in response to DOTS
clients requests for mitigation, and SHOULD accept scoped
mitigation requests from authorized DOTS clients. DOTS servers
MAY reject authorized requests for mitigation, but MUST include a
reason for the rejection in the status message sent to the client.

Due to the higher likelihood of packet loss during a DDoS attack,
DOTS servers SHOULD regularly send mitigation status to authorized
DOTS clients which have requested and been granted mitigation,
regardless of client requests for mitigation status.

When DOTS client-requested mitigation is active, DOTS server
status messages SHOULD include the following mitigation metrics:

* Total number of packets blocked by the mitigation

* Current number of packets per second blocked

* Total number of bytes blocked

* Current number of bytes per second blocked

DOTS clients MAY take these metrics into account when determining
whether to ask the DOTS server to cease mitigation.

Once a DOTS client requests mitigation, the client MAY withdraw
that request at any time, regardless of whether mitigation is
currently active. The DOTS server MUST immediately acknowledge a
DOTS client's request to stop mitigation.

To protect against route or DNS flapping caused by a client
rapidly toggling mitigation, and to dampen the effect of
oscillating attacks, DOTS servers MAY allow mitigation to continue
for a limited period after acknowledging a DOTS client's
withdrawal of a mitigation request. During this period, DOTS
server status messages SHOULD indicate that mitigation is active
but terminating.

The initial active-but-terminating period is implementation-
specific, but SHOULD be sufficiently long to absorb latency
incurred by route propagation. If the client requests mitigation
again before the initial active-but-terminating period elapses,
the DOTS server MAY exponentially increase the active-but-
terminating period up to a maximum of 300 seconds (5 minutes).
After the active-but-terminating period elapses, the DOTS server
MUST treat the mitigation as terminated, as the DOTS client is no
longer responsible for the mitigation. For example, if there is a
financial relationship between the DOTS client and server domains,
the DOTS client ceases incurring cost at this point.

SIG-006 Mitigation Lifetime: DOTS servers MUST support mitigation
   lifetimes, and MUST terminate a mitigation when the lifetime
   elapses. DOTS servers also MUST support renewal of mitigation
   lifetimes in mitigation requests from DOTS clients, allowing
   clients to extend mitigation as necessary for the duration of an
   attack.

   DOTS servers MUST treat a mitigation terminated due to lifetime
   expiration exactly as if the DOTS client originating the
   mitigation had asked to end the mitigation, including the active-
   but-terminating period, as described above in SIG-005.

   DOTS clients SHOULD include a mitigation lifetime in all
   mitigation requests. If a DOTS client does not include a
   mitigation lifetime in requests for help sent to the DOTS server,
   the DOTS server will use a reasonable default as defined by the
   protocol.

   DOTS servers SHOULD support indefinite mitigation lifetimes,
   enabling architectures in which the mitigator is always in the
   traffic path to the resources for which the DOTS client is
   requesting protection. DOTS clients MUST be prepared to not be
   granted mitigations with indefinite lifetimes. DOTS servers MAY

refuse mitigations with indefinite lifetimes, for policy reasons.
The reasons themselves are out of scope. If the DOTS server does
not grant a mitigation request with an indefinite mitigation
lifetime, it MUST set the lifetime to a value that is configured
locally. That value MUST be returned in a reply to the requesting
DOTS client.

SIG-007 Mitigation Scope: DOTS clients MUST indicate desired
mitigation scope. The scope type will vary depending on the
resources requiring mitigation. All DOTS agent implementations
MUST support the following required scope types:

* IPv4 addresses in dotted quad format

* IPv4 prefixes in CIDR notation [RFC4632]

* IPv6 addresses [RFC4291][RFC5952]

* IPv6 prefixes [RFC4291][RFC5952]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS agents MUST support mitigation scope aliases, allowing DOTS
clients and servers to refer to collections of protected resources
by an opaque identifier created through the data channel, direct
configuration, or other means. Domain name and URI mitigation
scopes may be thought of as a form of scope alias, in which the
addresses to which the domain name or URI resolve represent the
full scope of the mitigation.

If there is additional information available narrowing the scope
of any requested attack response, such as targeted port range,
protocol, or service, DOTS clients SHOULD include that information
in client signals. DOTS clients MAY also include additional
attack details. Such supplemental information is OPTIONAL, and
DOTS servers MAY ignore it when enabling countermeasures on the
mitigator.

As an active attack evolves, clients MUST be able to adjust as
necessary the scope of requested mitigation by refining the scope
of resources requiring mitigation.

The DOTS client may obtain the mitigation scope through direct
provisioning or through implementation-specific methods of

discovery. DOTS clients MUST support at least one mechanism to
obtain mitigiation scope.

SIG-008 Mitigation Efficacy: When a mitigation request by a DOTS
    client is active, DOTS clients SHOULD transmit a metric of
    perceived mitigation efficacy to the DOTS server. DOTS servers
    MAY use the efficacy metric to adjust countermeasures activated on
    a mitigator on behalf of a DOTS client.

SIG-009 Conflict Detection and Notification: Multiple DOTS clients
    controlled by a single administrative entity may send conflicting
    mitigation requests for pools of protected resources as a result
    of misconfiguration, operator error, or compromised DOTS clients.
    DOTS servers in the same administrative domain attempting to honor
    conflicting requests may flap network route or DNS information,
    degrading the networks attempting to participate in attack
    response with the DOTS clients. DOTS servers in a single
    administrative domain SHALL detect such conflicting requests, and
    SHALL notify the DOTS clients in conflict. The notification
    SHOULD indicate the nature and scope of the conflict, for example,
    the overlapping prefix range in a conflicting mitigation request.

SIG-010: Network Address Translator Traversal: DOTS clients may be
    deployed behind a Network Address Translator (NAT), and need to
    communicate with DOTS servers through the NAT. DOTS protocols
    MUST therefore be capable of traversing NATs.

    If UDP is used as the transport for the DOTS signal channel, all
    considerations in "Middlebox Traversal Guidelines" in [RFC8085]
    apply to DOTS. Regardless of transport, DOTS protocols MUST
    follow established best common practices (BCPs) for NAT traversal.

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges
between DOTS agents. Unlike the signal channel, which must operate
nominally even when confronted with signal degradation due to packet
loss, the data channel is not expected to be constructed to deal with
attack conditions. As the primary function of the data channel is
data exchange, a reliable transport is required in order for DOTS
agents to detect data delivery success or failure.

The data channel must be extensible. We anticipate the data channel
will be used for such purposes as configuration or resource
discovery. For example, a DOTS client may submit to the DOTS server
a collection of prefixes it wants to refer to by alias when
requesting mitigation, to which the server would respond with a
success status and the new prefix group alias, or an error status and

message in the event the DOTS client's data channel request failed.
The transactional nature of such data exchanges suggests a separate
set of requirements for the data channel, while the potentially
sensitive content sent between DOTS agents requires extra precautions
to ensure data privacy and authenticity.

DATA-001 Reliable transport: Messages sent over the data channel
   MUST be delivered reliably, in order sent.

DATA-002 Data privacy and integrity: Transmissions over the data
   channel are likely to contain operationally or privacy-sensitive
   information or instructions from the remote DOTS agent. Theft or
   modification of data channel transmissions could lead to
   information leaks or malicious transactions on behalf of the
   sending agent (see Section 4 below). Consequently data sent over
   the data channel MUST be encrypted and authenticated using current
   industry best practices. DOTS servers MUST enable means to
   prevent leaking operationally or privacy-sensitive data. Although
   administrative entities participating in DOTS may detail what data
   may be revealed to third-party DOTS agents, such considerations
   are not in scope for this document.

DATA-003 Resource Configuration: To help meet the general and signal
   channel requirements in this document, DOTS server implementations
   MUST provide an interface to configure resource identifiers, as
   described in SIG-007. DOTS server implementations MAY expose
   additional configurability. Additional configurability is
   implementation-specific.

DATA-004 Black- and whitelist management: DOTS servers MUST provide
   methods for DOTS clients to manage black- and white-lists of
   traffic destined for resources belonging to a client.

   For example, a DOTS client should be able to create a black- or
   whitelist entry; retrieve a list of current entries from either
   list; update the content of either list; and delete entries as
   necessary.

   How the DOTS server authorizes DOTS client management of black-
   and white-list entries is implementation-specific.

2.4. Security requirements

   DOTS must operate within a particularly strict security context, as
   an insufficiently protected signal or data channel may be subject to
   abuse, enabling or supplementing the very attacks DOTS purports to
   mitigate.

   SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate
      each other before a DOTS signal or data channel is considered
      valid. The method of authentication is not specified, but should
      follow current industry best practices with respect to any
      cryptographic mechanisms to authenticate the remote peer.

   SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS
      protocols MUST take steps to protect the confidentiality,
      integrity and authenticity of messages sent between client and
      server. While specific transport- and message-level security
      options are not specified, the protocols MUST follow current
      industry best practices for encryption and message authentication.

      In order for DOTS protocols to remain secure despite advancements
      in cryptanalysis and traffic analysis, DOTS agents MUST be able to
      negotiate the terms and mechanisms of protocol security, subject
      to the interoperability and signal message size requirements
      above.

      While the interfaces between downstream DOTS server and upstream
      DOTS client within a DOTS gateway are implementation-specific,
      those interfaces nevertheless MUST provide security equivalent to
      that of the signal channels bridged by gateways in the signaling
      path. For example, when a DOTS gateway consisting of a DOTS
      server and DOTS client is running on the same logical device, they
      must be within the same process security boundary.

   SEC-003 Message Replay Protection: To prevent a passive attacker
      from capturing and replaying old messages, and thereby potentially
      disrupting or influencing the network policy of the receiving DOTS
      agent's domain, DOTS protocols MUST provide a method for replay
      detection and prevention.

      Within the signal channel, messages MUST be uniquely identified
      such that replayed or duplicated messages may be detected and
      discarded. Unique mitigation requests MUST be processed at most
      once.

   SEC-004 Authorization: DOTS servers MUST authorize all messages from
      DOTS clients which pertain to mitigation, configuration,
      filtering, or status.

      DOTS servers MUST reject mitigation requests with scopes which the
      DOTS client is not authorized to manage.

      Likewise, DOTS servers MUST refuse to allow creation, modification
      or deletion of scope aliases and black-/white-lists when the DOTS
      client is unauthorized.

The modes of authorization are implementation-specific.

2.5. Data Model Requirements

   The value of DOTS is in standardizing a mechanism to permit elements,
   networks or domains under threat of DDoS attack to request aid
   mitigating the effects of any such attack. A well-structured DOTS
   data model is therefore critical to the development of a successful
   DOTS protocol.

   DM-001: Structure: The data model structure for the DOTS protocol
      may be described by a single module, or be divided into related
      collections of hierarchical modules and sub-modules. If the data
      model structure is split across modules, those distinct modules
      MUST allow references to describe the overall data model's
      structural dependencies.

   DM-002: Versioning: To ensure interoperability between DOTS protocol
      implementations, data models MUST be versioned. The version
      number of the initial data model SHALL be 1. Each published
      change to the initial published DOTS data model SHALL increment
      the data model version by 1.

      How the protocol represents data model versions is not defined in
      this document.

   DM-003: Mitigation Status Representation: The data model MUST
      provide the ability to represent a request for mitigation and the
      withdrawal of such a request. The data model MUST also support a
      representation of currently requested mitigation status, including
      failures and their causes.

   DM-004: Mitigation Scope Representation: The data model MUST support
      representation of a requested mitigation's scope. As mitigation
      scope may be represented in several different ways, per SIG-007
      above, the data model MUST be capable of flexible representation
      of mitigation scope.

   DM-005: Mitigation Lifetime Representation: The data model MUST
      support representation of a mitigation request's lifetime,
      including mitigations with no specified end time.

   DM-006: Mitigation Efficacy Representation: The data model MUST
      support representation of a DOTS client's understanding of the
      efficacy of a mitigation enabled through a mitigation request.

   DM-007: Acceptable Signal Loss Representation: The data model MUST
      be able to represent the DOTS agent's preference for acceptable

      signal loss when establishing a signal channel, as described in
      GEN-002.

   DM-008: Heartbeat Interval Representation: The data model MUST be
      able to represent the DOTS agent's preferred heartbeat interval,
      which the client may include when establishing the signal channel,
      as described in SIG-003.

   DM-009: Relationship to Transport: The DOTS data model MUST NOT
      depend on the specifics of any transport to represent fields in
      the model.

3. Congestion Control Considerations

3.1. Signal Channel

   As part of a protocol expected to operate over links affected by DDoS
   attack traffic, the DOTS signal channel MUST NOT contribute
   significantly to link congestion. To meet the signal channel
   requirements above, DOTS signal channel implementations SHOULD
   support connectionless transports. However, some connectionless
   transports when deployed naively can be a source of network
   congestion, as discussed in [RFC5405]. Signal channel
   implementations using such connectionless transports, such as UDP,
   therefore MUST include a congestion control mechanism.

   Signal channel implementations using TCP may rely on built-in TCP
   congestion control support.

3.2. Data Channel

   As specified in DATA-001, the data channel requires reliable, in-
   order message delivery. Data channel implementations using TCP may
   rely on the TCP implementation's built-in congestion control
   mechanisms.

4. Security Considerations

   DOTS is at risk from three primary attacks:

   o DOTS agent impersonation

   o Traffic injection

   o Signaling blocking

   The DOTS protocol MUST be designed for minimal data transfer to
   address the blocking risk. Impersonation and traffic injection

mitigation can be managed through current secure communications best
practices. See Section 2.4 above for a detailed discussion.

5. Contributors

    Mohamed Boucadair
       Orange

       mohamed.boucadair@orange.com

    Flemming Andreasen
       Cisco Systems, Inc.

       fandreas@cisco.com

    Dave Dolson
       Sandvine

       ddolson@sandvine.com

6. Acknowledgments

    Thanks to Roman Danyliw and Matt Richardson for careful reading and
    feedback.

7. Change Log

7.1. 04 revision

    2017-03-13

    o Establish required and optional mitigation scope types

    o Specify message size for DOTS signal channel

    o Recast mitigation lifetime as a DOTS server requirement

    o Clarify DOTS server's responsibilities after client request to end
       mitigation

    o Specify security state handling on redirection

    o Signal channel should use transport not susceptible to HOL
       blocking

    o Expanded list of DDoS types to include network links

7.2. 03 revision

   2016-10-30

   o Extended SEC-003 to require secure interfaces within DOTS
      gateways.

   o Changed DATA-003 to Resource Configuration, delegating control of
      acceptable signal loss, heartbeat intervals, and mitigation
      lifetime to DOTS client.

   o Added data model requirements reflecting client control over the
      above.

7.3. 02 revision

7.4. 01 revision

   2016-03-21

   o Reconciled terminology with -00 revision of
      [I-D.ietf-dots-use-cases].

   o Terminology clarification based on working group feedback.

   o Moved security-related requirements to separate section.

   o Made resilience/robustness primary general requirement to align
      with charter.

   o Clarified support for unidirectional communication within the
      bidirectional signal channel.

   o Added proposed operational requirement to support session
      redirection.

   o Added proposed operational requirement to support conflict
      notification.

   o Added proposed operational requirement to support mitigation
      lifetime in mitigation requests.

   o Added proposed operational requirement to support mitigation
      efficacy reporting from DOTS clients.

   o Added proposed operational requirement to cache lookups of all
      kinds.

   o Added proposed operational requirement regarding NAT traversal.

   o Removed redundant mutual authentication requirement from data
      channel requirements.

7.5. 00 revision

   2015-10-15

7.6. Initial revision

   2015-09-24 Andrew Mortensen

8. References

8.1. Normative References

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

   [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
              DOI 10.17487/RFC0791, September 1981,
              <https://www.rfc-editor.org/info/rfc791>.

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

   [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>.

   [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122,
              DOI 10.17487/RFC1122, October 1989,
              <https://www.rfc-editor.org/info/rfc1122>.

   [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
              DOI 10.17487/RFC1191, November 1990,
              <https://www.rfc-editor.org/info/rfc1191>.

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

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <https://www.rfc-editor.org/info/rfc3986>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
              (CIDR): The Internet Address Assignment and Aggregation
              Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
              2006, <https://www.rfc-editor.org/info/rfc4632>.

   [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
              Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007,
              <https://www.rfc-editor.org/info/rfc4821>.

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

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

   [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>.

8.2. Informative References

   [I-D.ietf-dots-architecture]
              Mortensen, A., Andreasen, F., Reddy, T.,
              christopher_gray3@cable.comcast.com, c., Compton, R., and
              N. Teague, "Distributed-Denial-of-Service Open Threat
              Signaling (DOTS) Architecture", draft-ietf-dots-
              architecture-04 (work in progress), July 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-07 (work
              in progress), July 2017.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <https://www.rfc-editor.org/info/rfc7092>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

Authors' Addresses

   Andrew Mortensen
   Arbor Networks
   2727 S. State St
   Ann Arbor, MI 48104
   United States

   Email: amortensen@arbor.net

   Robert Moskowitz
   Huawei
   Oak Park, MI 42837
   United States

   Email: rgm@htt-consult.com

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

   Email: TirumaleswarReddy_Konda@McAfee.com

DOTS   A. Mortensen
Internet-Draft   Arbor Networks
Intended status: Informational   T. Reddy
Expires: September 24, 2019   McAfee
                                                             R. Moskowitz
                                                                   Huawei
                                                           March 23, 2019

        Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
                      draft-ietf-dots-requirements-22

   Abstract

      This document defines the requirements for the Distributed Denial of
      Service (DDoS) Open Threat Signaling (DOTS) protocols enabling
      coordinated response to DDoS attacks.

   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 24, 2019.

the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

1.1. Context and Motivation

   Distributed Denial of Service (DDoS) attacks afflict networks
   connected to the Internet, plaguing network operators at service
   providers and enterprises around the world. High-volume attacks
   saturating inbound links are now common, as attack scale and
   frequency continue to increase.

   The prevalence and impact of these DDoS attacks has led to an
   increased focus on coordinated attack response. However, many
   enterprises lack the resources or expertise to operate on-premises
   attack mitigation solutions themselves, or are constrained by local
   bandwidth limitations. To address such gaps, service providers have
   begun to offer on-demand traffic scrubbing services, which are
   designed to separate the DDoS attack traffic from legitimate traffic
   and forward only the latter.

   Today, these services offer proprietary interfaces for subscribers to
   request attack mitigation. Such proprietary interfaces tie a
   subscriber to a service and limit the abilities of network elements

that would otherwise be capable of participating in attack
mitigation. As a result of signaling interface incompatibility,
attack responses may be fragmented or otherwise incomplete, leaving
operators in the attack path unable to assist in the defense.

A standardized method to coordinate a real-time response among
involved operators will increase the speed and effectiveness of DDoS
attack mitigation, and reduce the impact of these attacks. This
document describes the required characteristics of protocols that
enable attack response coordination and mitigation of DDoS attacks.

DDoS Open Threat Signaling (DOTS) communicates the need for defensive
action in anticipation of or in response to an attack, but does not
dictate the implementation of these actions. The DOTS use cases are
discussed in [I-D.ietf-dots-use-cases] and the DOTS architecture is
discussed in [I-D.ietf-dots-architecture].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in BCP14 [RFC2119]
[RFC8174], when, and only when, they appear in all capitals. These
capitalized words are used to signify the requirements for DOTS
protocols design.

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic
   originating from multiple sources is directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability and/or other functionality of an attack target.
   Denial-of-service considerations are discussed in detail in
   [RFC4732].

DDoS attack target: A network connected entity that is the target of
   a DDoS attack. Potential targets include (but are not limited to)
   network elements, network links, servers, and services.

DDoS attack telemetry: Collected measurements and behavioral
   characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions focused on recognizing
   and filtering out specific types of DDoS attack traffic while
   passing legitimate traffic to the attack target. Distinct
   countermeasures can be layered to defend against attacks combining
   multiple DDoS attack types.

Mitigation: A set of countermeasures enforced against traffic
   destined for the target or targets of a detected or reported DDoS
   attack, where countermeasure enforcement is managed by an entity
   in the network path between attack sources and the attack target.
   Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of
   performing mitigation of a detected or reported DDoS attack. The
   means by which this entity performs these mitigations and how they
   are requested of it are out of scope for this document. The
   mitigator and DOTS server receiving a mitigation request are
   assumed to belong to the same administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting
   attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to
   messages from DOTS clients. The DOTS server enables mitigation on
   behalf of the DOTS client, if requested, by communicating the DOTS
   client's request to the mitigator and returning selected mitigator
   feedback to the requesting DOTS client.

DOTS agent: Any DOTS-aware software module capable of participating
   in a DOTS signal or data channel. It can be a DOTS client, DOTS
   server, or, as a logical agent, a DOTS gateway.

DOTS gateway: A DOTS-aware software module resulting from the
   logical concatenation of the functionality of a DOTS server and a
   DOTS client into a single DOTS agent. This functionality is
   analogous to a Session Initiation Protocol (SIP) [RFC3261] Back-
   to-Back User Agent (B2BUA) [RFC7092]. A DOTS gateway has a
   client-facing side, which behaves as a DOTS server for downstream
   clients, and a server-facing side, which performs the role of DOTS
   client for upstream DOTS servers. Client-domain DOTS gateways are
   DOTS gateways that are in the DOTS client's domain, while server-
   domain DOTS gateways denote DOTS gateways that are in the DOTS
   server's domain. A DOTS gateway may terminate multiple discrete
   DOTS client connections and may aggregate these into a single or
   multiple connections. DOTS gateways are described further in
   [I-D.ietf-dots-architecture].

Signal channel: A bidirectional, mutually authenticated
   communication channel between DOTS agents that is resilient even
   in conditions leading to severe packet loss, such as a volumetric
   DDoS attack causing network congestion.

DOTS signal: A status/control message transmitted over the
   authenticated signal channel between DOTS agents, used to indicate

the client's need for mitigation, or to convey the status of any
requested mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal
   channel, used as a keep-alive and to measure peer health.

Data channel: A bidirectional, mutually authenticated communication
   channel between two DOTS agents used for infrequent but reliable
   bulk exchange of data not easily or appropriately communicated
   through the signal channel. Reliable bulk data exchange may not
   function well or at all during attacks causing network congestion.
   The data channel is not expected to operate in such conditions.

Filter: A specification of a matching network traffic flow or set of
   flows. The filter will typically have a policy associated with
   it, e.g., rate-limiting or discarding matching traffic [RFC4949].

Drop-list: A list of filters indicating sources from which traffic
   should be blocked, regardless of traffic content.

Accept-list: A list of filters indicating sources from which traffic
   should always be allowed, regardless of contradictory data gleaned
   in a detected attack.

Multi-homed DOTS client: A DOTS client exchanging messages with
   multiple DOTS servers, each in a separate administrative domain.

2. Requirements

   The expected layout and interactions amongst DOTS entities is
   described in the DOTS Architecture [I-D.ietf-dots-architecture].

   The goal of the DOTS requirements specification is to specify the
   requirements for DOTS signal channel and data channel protocols that
   have different application and transport layer requirements. This
   section describes the required features and characteristics of the
   DOTS protocols.

   The goal of DOTS protocols is to enable and manage mitigation on
   behalf of a network domain or resource which is or may become the
   focus of a DDoS attack. An active DDoS attack against the entity
   controlling the DOTS client need not be present before establishing a
   communication channel between DOTS agents. Indeed, establishing a
   relationship with peer DOTS agents during normal network conditions
   provides the foundation for more rapid attack response against future
   attacks, as all interactions setting up DOTS, including any business
   or service level agreements, are already complete. Reachability
   information of peer DOTS agents is provisioned to a DOTS client using

a variety of manual or dynamic methods. Once a relationship between
DOTS agents is established, regular communication between DOTS
clients and servers enables a common understanding of the DOTS
agents' health and activity.

The DOTS protocol must at a minimum make it possible for a DOTS
client to request aid mounting a defense against a suspected attack.
This defense could be coordinated by a DOTS server and include
signaling within or between domains as requested by local operators.
DOTS clients should similarly be able to withdraw aid requests. DOTS
requires no justification from DOTS clients for requests for help,
nor do DOTS clients need to justify withdrawing help requests: the
decision is local to the DOTS clients' domain. Multi-homed DOTS
clients must be able to select the appropriate DOTS server(s) to
which a mitigation request is to be sent. The method for selecting
the appropriate DOTS server in a multi-homed environment is out of
scope for this document.

DOTS protocol implementations face competing operational goals when
maintaining this bidirectional communication stream. On the one
hand, DOTS must include measures to ensure message confidentiality,
integrity, authenticity, and replay protection to keep the protocols
from becoming additional vectors for the very attacks it is meant to
help fight off. On the other hand, the protocol must be resilient
under extremely hostile network conditions, providing continued
contact between DOTS agents even as attack traffic saturates the
link. Such resiliency may be developed several ways, but
characteristics such as small message size, asynchronous, redundant
message delivery and minimal connection overhead (when possible given
local network policy) will tend to contribute to the robustness
demanded by a viable DOTS protocol. Operators of peer DOTS-enabled
domains may enable quality- or class-of-service traffic tagging to
increase the probability of successful DOTS signal delivery, but DOTS
does not require such policies be in place, and should be viable in
their absence.

The DOTS server and client must also have some standardized method of
defining the scope of any mitigation, as well as managing other
mitigation-related configuration.

Finally, DOTS should be sufficiently extensible to meet future needs
in coordinated attack defense, although this consideration is
necessarily superseded by the other operational requirements.

2.1. General Requirements

   GEN-001 Extensibility: Protocols and data models developed as part
      of DOTS MUST be extensible in order to keep DOTS adaptable to
      proprietary DDoS defenses. Future extensions MUST be backward
      compatible. Implementations of older protocol versions MUST
      ignore optional information added to DOTS messages as part of
      newer protocol versions. Implementations of older protocol
      versions MUST reject DOTS messages carrying mandatory information
      as part of newer protocol versions.

   GEN-002 Resilience and Robustness: The signaling protocol MUST be
      designed to maximize the probability of signal delivery even under
      the severely constrained network conditions caused by attack
      traffic. Additional means to enhance the resilience of DOTS
      protocols, including when multiple DOTS servers are provisioned to
      the DOTS clients, SHOULD be considered. The protocol MUST be
      resilient, that is, continue operating despite message loss and
      out-of-order or redundant message delivery. In support of
      signaling protocol robustness, DOTS signals SHOULD be conveyed
      over transport and application protocols not susceptible to Head
      of Line Blocking. These requirements are at SHOULD strength to
      handle middle-boxes and firewall traversal.

   GEN-003 Bulk Data Exchange: Infrequent bulk data exchange between
      DOTS agents can also significantly augment attack response
      coordination, permitting such tasks as population of drop- or
      accept-listed source addresses; address or prefix group aliasing;
      exchange of incident reports; and other hinting or configuration
      supplementing attack response.

      As the resilience requirements for the DOTS signal channel mandate
      small signal message size, a separate, secure data channel
      utilizing a reliable transport protocol MUST be used for bulk data
      exchange. However, reliable bulk data exchange may not be
      possible during attacks causing network congestion.

   GEN-004 Mitigation Hinting: DOTS clients may have access to attack
      details which can be used to inform mitigation techniques.
      Example attack details might include locally collected
      fingerprints for an on-going attack, or anticipated or active
      attack focal points based on other threat intelligence. DOTS
      clients MAY send mitigation hints derived from attack details to
      DOTS servers, in the full understanding that the DOTS server MAY
      ignore mitigation hints. Mitigation hints MUST be transmitted
      across the signal channel, as the data channel may not be
      functional during an attack. DOTS server handling of mitigation
      hints is implementation-specific.

GEN-005 Loop Handling: In certain scenarios, typically involving
   misconfiguration of DNS or routing policy, it may be possible for
   communication between DOTS agents to loop. Signal and data
   channel implementations should be prepared to detect and terminate
   such loops to prevent service disruption.

2.2. Signal Channel Requirements

SIG-001 Use of Common Transport Protocols: DOTS MUST operate over
   common widely deployed and standardized transport protocols.
   While connectionless transport such as the User Datagram Protocol
   (UDP) [RFC0768] SHOULD be used for the signal channel, the
   Transmission Control Protocol (TCP) [RFC0793] MAY be used if
   necessary due to network policy or middlebox capabilities or
   configurations.

SIG-002 Sub-MTU Message Size: To avoid message fragmentation and the
   consequently decreased probability of message delivery over a
   congested link, signaling protocol message size MUST be kept under
   signaling Path Maximum Transmission Unit (PMTU), including the
   byte overhead of any encapsulation, transport headers, and
   transport- or message-level security. If the total message size
   exceeds the path MTU, the DOTS agent MUST split the message into
   separate messages; for example, the list of mitigation scope types
   could be split into multiple lists and each list conveyed in a new
   message.

   DOTS agents can attempt to learn PMTU using the procedures
   discussed in [I-D.ietf-intarea-frag-fragile]. If the PMTU cannot
   be discovered, DOTS agents MUST assume a PMTU of 1280 bytes, as
   IPv6 requires that every link in the Internet have an MTU of 1280
   octets or greater as specified in [RFC8200]. If IPv4 support on
   legacy or otherwise unusual networks is a consideration and the
   PMTU is unknown, DOTS implementations MAY assume on a PMTU of 576
   bytes for IPv4 datagrams, as every IPv4 host must be capable of
   receiving a packet whose length is equal to 576 bytes as discussed
   in [RFC0791] and [RFC1122].

SIG-003 Bidirectionality: To support peer health detection, to
   maintain an active signal channel, and to increase the probability
   of signal delivery during an attack, the signal channel MUST be
   bidirectional, with client and server transmitting signals to each
   other at regular intervals, regardless of any client request for
   mitigation. The bidirectional signal channel MUST support
   unidirectional messaging to enable notifications between DOTS
   agents.

SIG-004 Channel Health Monitoring: DOTS agents MUST support exchange
   of heartbeat messages over the signal channel to monitor channel
   health. These keepalives serve the purpose to maintain any on-
   path NAT or Firewall bindings to avoid cryptographic handshake for
   new mitigation requests. The heartbeat interval during active
   mitigation could be negotiable based on NAT/Firewall
   characteristics. Absent information about the NAT/Firewall
   characteristics, DOTS agent needs to ensure its on-path NAT or
   Firewall bindings do not expire, by using the keep-alive frequency
   discussed in Section 3.5 of [RFC8085].

   To support scenarios in which loss of heartbeat is used to trigger
   mitigation, and to keep the channel active, DOTS servers MUST
   solicit heartbeat exchanges after successful mutual
   authentication. When DOTS agents are exchanging heartbeats and no
   mitigation request is active, either agent MAY request changes to
   the heartbeat rate. For example, a DOTS server might want to
   reduce heartbeat frequency or cease heartbeat exchanges when an
   active DOTS client has not requested mitigation, in order to
   control load.

   Following mutual authentication, a signal channel MUST be
   considered active until a DOTS agent explicitly ends the session.
   When no attack traffic is present, the signal channel MUST be
   considered active until either DOTS agent fails to receive
   heartbeats from the other peer after a mutually agreed upon
   retransmission procedure has been exhausted. Peer DOTS agents
   MUST regularly send heartbeats to each other while a mitigation
   request is active. Because heartbeat loss is much more likely
   during volumetric attack, DOTS agents SHOULD avoid signal channel
   termination when mitigation is active and heartbeats are not
   received by either DOTS agent for an extended period. The
   exception circumstances to terminate the signal channel session
   during active mitigation are discussed below:

   * To handle possible DOTS server restart or crash, the DOTS
      clients MAY attempt to establish a new signal channel session,
      but MUST continue to send heartbeats on the current session so
      that the DOTS server knows the session is still alive. If the
      new session is successfully established, the DOTS client can
      terminate the current session.

   * DOTS servers are assumed to have the ability to monitor the
      attack, using feedback from the mitigator and other available
      sources, and MAY use the absence of attack traffic and lack of

client heartbeats as an indication the signal channel is
defunct.

SIG-005 Channel Redirection: In order to increase DOTS operational
flexibility and scalability, DOTS servers SHOULD be able to
redirect DOTS clients to another DOTS server at any time. DOTS
clients MUST NOT assume the redirection target DOTS server shares
security state with the redirecting DOTS server. DOTS clients are
free to attempt abbreviated security negotiation methods supported
by the protocol, such as DTLS session resumption, but MUST be
prepared to negotiate new security state with the redirection
target DOTS server. The redirection DOTS server and redirecting
DOTS server MUST belong to the same administrative domain.

Due to the increased likelihood of packet loss caused by link
congestion during an attack, DOTS servers SHOULD NOT redirect
while mitigation is enabled during an active attack against a
target in the DOTS client's domain.

SIG-006 Mitigation Requests and Status: Authorized DOTS clients MUST
be able to request scoped mitigation from DOTS servers. DOTS
servers MUST send status to the DOTS clients about mitigation
requests. If a DOTS server rejects an authorized request for
mitigation, the DOTS server MUST include a reason for the
rejection in the status message sent to the client.

DOTS servers MUST regularly send mitigation status updates to
authorized DOTS clients which have requested and been granted
mitigation. If unreliable transport is used for the signal
channel protocol, due to the higher likelihood of packet loss
during a DDoS attack, DOTS servers needs to send mitigation status
multiple times at regular intervals following the the data
transmission guidelines discussed in Section 3.1.3 of [RFC8085].

When DOTS client-requested mitigation is active, DOTS server
status messages MUST include the following mitigation metrics:

* Total number of packets blocked by the mitigation

* Current number of packets per second blocked

* Total number of bytes blocked

* Current number of bytes per second blocked

DOTS clients MAY take these metrics into account when determining
whether to ask the DOTS server to cease mitigation.

A DOTS client MAY withdraw a mitigation request at any time,
regardless of whether mitigation is currently active. The DOTS
server MUST immediately acknowledge a DOTS client's request to
stop mitigation.

To protect against route or DNS flapping caused by a client
rapidly toggling mitigation, and to dampen the effect of
oscillating attacks, DOTS servers MAY allow mitigation to continue
for a limited period after acknowledging a DOTS client's
withdrawal of a mitigation request. During this period, DOTS
server status messages SHOULD indicate that mitigation is active
but terminating. DOTS clients MAY reverse the mitigation
termination during this active-but-terminating period with a new
mitigation request for the same scope. The DOTS server MUST treat
this request as a mitigation lifetime extension (see SIG-007
below).

The initial active-but-terminating period is implementation- and
deployment- specific, but SHOULD be sufficiently long to absorb
latency incurred by route propagation. If a DOTS client refreshes
the mitigation before the active-but-terminating period elapses,
the DOTS server MAY increase the active-but-terminating period up
to a maximum of 300 seconds (5 minutes). After the active-but-
terminating period elapses, the DOTS server MUST treat the
mitigation as terminated, as the DOTS client is no longer
responsible for the mitigation.

SIG-007 Mitigation Lifetime: DOTS servers MUST support mitigations
   for a negotiated time interval, and MUST terminate a mitigation
   when the lifetime elapses. DOTS servers also MUST support renewal
   of mitigation lifetimes in mitigation requests from DOTS clients,
   allowing clients to extend mitigation as necessary for the
   duration of an attack.

   DOTS servers MUST treat a mitigation terminated due to lifetime
   expiration exactly as if the DOTS client originating the
   mitigation had asked to end the mitigation, including the active-
   but-terminating period, as described above in SIG-005.

   DOTS clients MUST include a mitigation lifetime in all mitigation
   requests.

   DOTS servers SHOULD support indefinite mitigation lifetimes,
   enabling architectures in which the mitigator is always in the
   traffic path to the resources for which the DOTS client is
   requesting protection. DOTS clients MUST be prepared to not be
   granted mitigations with indefinite lifetimes. DOTS servers MAY
   refuse mitigations with indefinite lifetimes, for policy reasons.

The reasons themselves are out of scope. If the DOTS server does
not grant a mitigation request with an indefinite mitigation
lifetime, it MUST set the lifetime to a value that is configured
locally. That value MUST be returned in a reply to the requesting
DOTS client.

SIG-008 Mitigation Scope: DOTS clients MUST indicate desired
mitigation scope. The scope type will vary depending on the
resources requiring mitigation. All DOTS agent implementations
MUST support the following required scope types:

* IPv4 prefixes [RFC4632]

* IPv6 prefixes [RFC4291][RFC5952]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS servers MUST be able to resolve domain names and (when
supported) URIs. How name resolution is managed on the DOTS
server is implementation-specific.

DOTS agents MUST support mitigation scope aliases, allowing DOTS
clients and servers to refer to collections of protected resources
by an opaque identifier created through the data channel, direct
configuration, or other means. Domain name and URI mitigation
scopes may be thought of as a form of scope alias, in which the
addresses to which the domain name or URI resolve represent the
full scope of the mitigation.

If there is additional information available narrowing the scope
of any requested attack response, such as targeted port range,
protocol, or service, DOTS clients SHOULD include that information
in client mitigation requests. DOTS clients MAY also include
additional attack details. DOTS servers MAY ignore such
supplemental information when enabling countermeasures on the
mitigator.

As an active attack evolves, DOTS clients MUST be able to adjust
as necessary the scope of requested mitigation by refining the
scope of resources requiring mitigation.

A DOTS client may obtain the mitigation scope through direct
provisioning or through implementation-specific methods of

discovery. DOTS clients MUST support at least one mechanism to
obtain mitigation scope.

SIG-009 Mitigation Efficacy: When a mitigation request is active,
DOTS clients MUST be able to transmit a metric of perceived
mitigation efficacy to the DOTS server. DOTS servers MAY use the
efficacy metric to adjust countermeasures activated on a mitigator
on behalf of a DOTS client.

SIG-010 Conflict Detection and Notification: Multiple DOTS clients
controlled by a single administrative entity may send conflicting
mitigation requests as a result of misconfiguration, operator
error, or compromised DOTS clients. DOTS servers in the same
administrative domain attempting to honor conflicting requests may
flap network route or DNS information, degrading the networks
attempting to participate in attack response with the DOTS
clients. DOTS servers in a single administrative domain SHALL
detect such conflicting requests, and SHALL notify the DOTS
clients in conflict. The notification MUST indicate the nature
and scope of the conflict, for example, the overlapping prefix
range in a conflicting mitigation request.

SIG-011 Network Address Translator Traversal: DOTS clients may be
deployed behind a Network Address Translator (NAT), and need to
communicate with DOTS servers through the NAT. DOTS protocols
MUST therefore be capable of traversing NATs.

If UDP is used as the transport for the DOTS signal channel, all
considerations in "Middlebox Traversal Guidelines" in [RFC8085]
apply to DOTS. Regardless of transport, DOTS protocols MUST
follow established best common practices established in BCP 127
for NAT traversal [RFC4787][RFC6888][RFC7857].

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges
between DOTS agents. Unlike the signal channel, the data channel is
not expected to be constructed to deal with attack conditions. As
the primary function of the data channel is data exchange, a reliable
transport is required in order for DOTS agents to detect data
delivery success or failure.

The data channel provides a protocol for DOTS configuration,
management. For example, a DOTS client may submit to a DOTS server a
collection of prefixes it wants to refer to by alias when requesting
mitigation, to which the server would respond with a success status
and the new prefix group alias, or an error status and message in the
event the DOTS client's data channel request failed.

DATA-001 Reliable transport: Messages sent over the data channel
          MUST be delivered reliably, in order sent.

DATA-003 Resource Configuration: To help meet the general and signal
          channel requirements in Section 2.1 and Section 2.2, DOTS server
          implementations MUST provide an interface to configure resource
          identifiers, as described in SIG-008. DOTS server implementations
          MAY expose additional configurability. Additional configurability
          is implementation-specific.

DATA-004 Policy management: DOTS servers MUST provide methods for
          DOTS clients to manage drop- and accept-lists of traffic destined
          for resources belonging to a client.

          For example, a DOTS client should be able to create a drop- or
          accept-list entry, retrieve a list of current entries from either
          list, update the content of either list, and delete entries as
          necessary.

          How a DOTS server authorizes DOTS client management of drop- and
          accept-list entries is implementation-specific.

2.4. Security Requirements

   DOTS must operate within a particularly strict security context, as
   an insufficiently protected signal or data channel may be subject to
   abuse, enabling or supplementing the very attacks DOTS purports to
   mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate
         each other before a DOTS signal or data channel is considered
         valid. The method of authentication is not specified in this
         document, but should follow current IETF best practices [RFC7525]
         with respect to any cryptographic mechanisms to authenticate the
         remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS
         protocols MUST take steps to protect the confidentiality,
         integrity and authenticity of messages sent between client and
         server. While specific transport- and message-level security
         options are not specified, the protocols MUST follow current IETF
         best practices [RFC7525] for encryption and message
         authentication. Client-domain DOTS gateways are more trusted than
         DOTS clients, while server-domain DOTS gateways and DOTS servers
         share the same level of trust. A security mechanism at the
         transport layer TLS/DTLS is thus adequate to provide security
         between peer DOTS agents.

In order for DOTS protocols to remain secure despite advancements in cryptanalysis and traffic analysis, DOTS agents MUST support secure negotiation of the terms and mechanisms of protocol security, subject to the interoperability and signal message size requirements in Section 2.2.

While the interfaces between downstream DOTS server and upstream DOTS client within a DOTS gateway are implementation-specific, those interfaces nevertheless MUST provide security equivalent to that of the signal channels bridged by gateways in the signaling path. For example, when a DOTS gateway consisting of a DOTS server and DOTS client is running on the same logical device, the two DOTS agents could be implemented within the same process security boundary.

SEC-003 Data privacy and integrity: Transmissions over the DOTS protocols are likely to contain operationally or privacy-sensitive information or instructions from the remote DOTS agent. Theft, modification, or replay of message transmissions could lead to information leaks or malicious transactions on behalf of the sending agent (see Section 4 below). Consequently data sent over the DOTS protocols MUST be encrypted using secure transports TLS/ DTLS. DOTS servers MUST enable means to prevent leaking operationally or privacy-sensitive data. Although administrative entities participating in DOTS may detail what data may be revealed to third-party DOTS agents, such considerations are not in scope for this document.

SEC-004 Message Replay Protection: To prevent a passive attacker from capturing and replaying old messages, and thereby potentially disrupting or influencing the network policy of the receiving DOTS agent's domain, DOTS protocols MUST provide a method for replay detection and prevention.

Within the signal channel, messages MUST be uniquely identified such that replayed or duplicated messages can be detected and discarded. Unique mitigation requests MUST be processed at most once.

SEC-005 Authorization: DOTS servers MUST authorize all messages from DOTS clients which pertain to mitigation, configuration, filtering, or status.

DOTS servers MUST reject mitigation requests with scopes which the DOTS client is not authorized to manage.

Likewise, DOTS servers MUST refuse to allow creation, modification or deletion of scope aliases and drop-/accept-lists when the DOTS client is unauthorized.

The modes of authorization are implementation-specific.

2.5. Data Model Requirements

A well-structured DOTS data model is critical to the development of successful DOTS protocols.

DM-001 Structure: The data model structure for the DOTS protocol MAY be described by a single module, or be divided into related collections of hierarchical modules and sub-modules. If the data model structure is split across modules, those distinct modules MUST allow references to describe the overall data model's structural dependencies.

DM-002 Versioning: To ensure interoperability between DOTS protocol implementations, data models MUST be versioned. How the protocols represent data model versions is not defined in this document.

DM-003 Mitigation Status Representation: The data model MUST provide the ability to represent a request for mitigation and the withdrawal of such a request. The data model MUST also support a representation of currently requested mitigation status, including failures and their causes.

DM-004 Mitigation Scope Representation: The data model MUST support representation of a requested mitigation's scope. As mitigation scope may be represented in several different ways, per SIG-008 above, the data model MUST include extensible representation of mitigation scope.

DM-005 Mitigation Lifetime Representation: The data model MUST support representation of a mitigation request's lifetime, including mitigations with no specified end time.

DM-006 Mitigation Efficacy Representation: The data model MUST support representation of a DOTS client's understanding of the efficacy of a mitigation enabled through a mitigation request.

DM-007 Acceptable Signal Loss Representation: The data model MUST be able to represent the DOTS agent's preference for acceptable signal loss when establishing a signal channel. Measurements of loss might include, but are not restricted to, number of consecutive missed heartbeat messages, retransmission count, or request timeouts.

   DM-008 Heartbeat Interval Representation: The data model MUST be
      able to represent the DOTS agent's preferred heartbeat interval,
      which the client may include when establishing the signal channel,
      as described in SIG-003.

   DM-009 Relationship to Transport: The DOTS data model MUST NOT make
      any assumptions about specific characteristics of any given
      transport into the data model, but instead, represent the fields
      in the model explicitly.

3. Congestion Control Considerations

3.1. Signal Channel

   As part of a protocol expected to operate over links affected by DDoS
   attack traffic, the DOTS signal channel MUST NOT contribute
   significantly to link congestion. To meet the signal channel
   requirements above, DOTS signal channel implementations SHOULD
   support connectionless transports. However, some connectionless
   transports when deployed naively can be a source of network
   congestion, as discussed in [RFC8085]. Signal channel
   implementations using such connectionless transports, such as UDP,
   therefore MUST include a congestion control mechanism.

   Signal channel implementations using a IETF standard congestion-
   controlled transport protocol (like TCP) may rely on built-in
   transport congestion control support.

3.2. Data Channel

   As specified in DATA-001, the data channel requires reliable, in-
   order message delivery. Data channel implementations using a IETF
   standard congestion-controlled transport protocol may rely on the
   transport implementation's built-in congestion control mechanisms.

4. Security Considerations

   This document informs future protocols under development, and so does
   not have security considerations of its own. However, operators
   should be aware of potential risks involved in deploying DOTS. DOTS
   agent impersonation and signal blocking are discussed here.
   Additional DOTS security considerations may be found in
   [I-D.ietf-dots-architecture] and DOTS protocol documents.

   Impersonation of either a DOTS server or a DOTS client could have
   catastrophic impact on operations in either domain. If an attacker
   has the ability to impersonate a DOTS client, that attacker can
   affect policy on the network path to the DOTS client's domain, up to

and including instantiation of drop-lists blocking all inbound
traffic to networks for which the DOTS client is authorized to
request mitigation.

Similarly, an impersonated DOTS server may be able to act as a sort
of malicious DOTS gateway, intercepting requests from the downstream
DOTS client, and modifying them before transmission to the DOTS
server to inflict the desired impact on traffic to or from the DOTS
client's domain. Among other things, this malicious DOTS gateway
might receive and discard mitigation requests from the DOTS client,
ensuring no requested mitigation is ever applied.

To detect misuse, as detailed in Section 2.4, DOTS implementations
require mutual authentication of DOTS agents in order to make agent
impersonation more difficult. However, impersonation may still be
possible as a result of credential theft, implementation flaws, or
compromise of DOTS agents.

To detect compromised DOTS agents, DOTS operators should carefully
monitor and audit DOTS agents to detect misbehavior and to deter
misuse, while employing current secure network communications best
practices to reduce attack surface.

Blocking communication between DOTS agents has the potential to
disrupt the core function of DOTS, which is to request mitigation of
active or expected DDoS attacks. The DOTS signal channel is expected
to operate over congested inbound links, and, as described in
Section 2.2, the signal channel protocol must be designed for minimal
data transfer to reduce the incidence of signal loss.

5. IANA Considerations

   This document does not require any IANA action.

6. Contributors

   Mohamed Boucadair
      Orange

      mohamed.boucadair@orange.com

   Flemming Andreasen
      Cisco Systems, Inc.

      fandreas@cisco.com

   Dave Dolson
      Sandvine

      ddolson@sandvine.com

7. Acknowledgments

   Thanks to Roman Danyliw, Matt Richardson, Joe Touch, Scott Bradner,
   Robert Sparks, Brian Weis, Benjamin Kaduk, Eric Rescorla, Alvaro
   Retana, Suresh Krishnan, Ben Campbell, Mirja Kuehlewind and Jon
   Shallow for careful reading and feedback.

8. References

8.1. Normative References

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

   [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
              DOI 10.17487/RFC0791, September 1981,
              <https://www.rfc-editor.org/info/rfc791>.

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

   [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>.

   [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122,
              DOI 10.17487/RFC1122, October 1989,
              <https://www.rfc-editor.org/info/rfc1122>.

   [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>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <https://www.rfc-editor.org/info/rfc3986>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
              (CIDR): The Internet Address Assignment and Aggregation
              Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
              2006, <https://www.rfc-editor.org/info/rfc4632>.

   [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address
              Translation (NAT) Behavioral Requirements for Unicast
              UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
              2007, <https://www.rfc-editor.org/info/rfc4787>.

   [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>.

   [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa,
              A., and H. Ashida, "Common Requirements for Carrier-Grade
              NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888,
              April 2013, <https://www.rfc-editor.org/info/rfc6888>.

   [RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar,
              S., and K. Naito, "Updates to Network Address Translation
              (NAT) Behavioral Requirements", BCP 127, RFC 7857,
              DOI 10.17487/RFC7857, April 2016,
              <https://www.rfc-editor.org/info/rfc7857>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

   [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>.

   [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", STD 86, RFC 8200,
              DOI 10.17487/RFC8200, July 2017,
              <https://www.rfc-editor.org/info/rfc8200>.

8.2. Informative References

   [I-D.ietf-dots-architecture]
              Mortensen, A., Andreasen, F., K, R., Teague, N., Compton,
              R., and c. christopher_gray3@cable.comcast.com,
              "Distributed-Denial-of-Service Open Threat Signaling
              (DOTS) Architecture", draft-ietf-dots-architecture-12
              (work in progress), February 2019.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-17 (work
              in progress), January 2019.

   [I-D.ietf-intarea-frag-fragile]
              Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O.,
              and F. Gont, "IP Fragmentation Considered Fragile", draft-
              ietf-intarea-frag-fragile-09 (work in progress), February
              2019.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <https://www.rfc-editor.org/info/rfc7092>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
              FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
              <https://www.rfc-editor.org/info/rfc4949>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

Authors' Addresses

   Andrew Mortensen
   Arbor Networks
   2727 S. State St
   Ann Arbor, MI 48104
   United States

   Email: andrewmortensen@gmail.com

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

      Email: TirumaleswarReddy_Konda@McAfee.com

      Robert Moskowitz
      Huawei
      Oak Park, MI 42837
      United States

      Email: rgm@htt-consult.com

DOTS   T. Reddy
Internet-Draft   McAfee
Intended status: Standards Track   M. Boucadair
Expires: May 16, 2018   Orange
                                                             P. Patil
                                                                Cisco
                                                         A. Mortensen
                                                 Arbor Networks, Inc.
                                                            N. Teague
                                                       Verisign, Inc.
                                                    November 12, 2017

       Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
                                   Channel
                    draft-ietf-dots-signal-channel-07

Abstract

   This document specifies the DOTS signal channel, a protocol for
   signaling the need for protection against Distributed Denial-of-
   Service (DDoS) attacks to a server capable of enabling network
   traffic mitigation on behalf of the requesting client. A companion
   document defines the DOTS data channel, a separate reliable
   communication layer for DOTS management and configuration.

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 16, 2018.

Table of Contents

1. Introduction

   A distributed denial-of-service (DDoS) attack is an attempt to make
   machines or network resources unavailable to their intended users.
   In most cases, sufficient scale can be achieved by compromising
   enough end-hosts and using those infected hosts to perpetrate and
   amplify the attack. The victim in this attack can be an application
   server, a host, a router, a firewall, or an entire network.

   In many cases, it may not be possible for network administrators to
   determine the causes of an attack, but instead just realize that
   certain resources seem to be under attack. This document defines a
   lightweight protocol permitting a DOTS client to request mitigation
   from one or more DOTS servers for protection against detected,
   suspected, or anticipated attacks . This protocol enables cooperation
   between DOTS agents to permit a highly-automated network defense that
   is robust, reliable and secure.

   The document adheres to the DOTS architecture
   [I-D.ietf-dots-architecture]. The requirements for DOTS signal
   channel protocol are obtained from [I-D.ietf-dots-requirements].
   This document satisfies all the use cases discussed in
   [I-D.ietf-dots-use-cases].

   This is a companion document to the DOTS data channel specification
   [I-D.ietf-dots-data-channel] that defines a configuration and bulk
   data exchange mechanism supporting the DOTS signal channel.

2. Notational Conventions and 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
   [RFC2119].

   (D)TLS: For brevity this term is used for statements that apply to
   both Transport Layer Security [RFC5246] and Datagram Transport Layer

Security [RFC6347]. Specific terms will be used for any statement
that applies to either protocol alone.

The reader should be familiar with the terms defined in
[I-D.ietf-dots-architecture].

3. Solution Overview

Network applications have finite resources like CPU cycles, number of
processes or threads they can create and use, maximum number of
simultaneous connections it can handle, limited resources of the
control plane, etc. When processing network traffic, such
applications are supposed to use these resources to offer the
intended task in the most efficient fashion. However, an attacker
may be able to prevent an application from performing its intended
task by causing the application to exhaust the finite supply of a
specific resource.

TCP DDoS SYN-flood, for example, is a memory-exhaustion attack on the
victim and ACK-flood is a CPU exhaustion attack on the victim
([RFC4987]). Attacks on the link are carried out by sending enough
traffic such that the link becomes excessively congested, and
legitimate traffic suffers high packet loss. Stateful firewalls can
also be attacked by sending traffic that causes the firewall to hold
excessive state. The firewall then runs out of memory, and can no
longer instantiate the state required to pass legitimate flows.
Other possible DDoS attacks are discussed in [RFC4732].

In each of the cases described above, the possible arrangements
between the DOTS client and DOTS server to mitigate the attack are
discussed in [I-D.ietf-dots-use-cases]. An example of network
diagram showing a deployment of these elements is shown in Figure 1.
Architectural relationships between involved DOTS agents is explained
in [I-D.ietf-dots-architecture]. In this example, the DOTS server is
operating on the access network.

```
   Network
   Resource              CPE router           Access network      _____
 +-----------+        +--------------+        +-------------+     /          \
 |           |        |              |        |             |    /            \
 |DOTS client|--------| DOTS gateway |--------| DOTS server |___ |  Internet  |
 |           |        |              |        |             |    |            |
 +-----------+        +--------------+        +-------------+     _____/
```
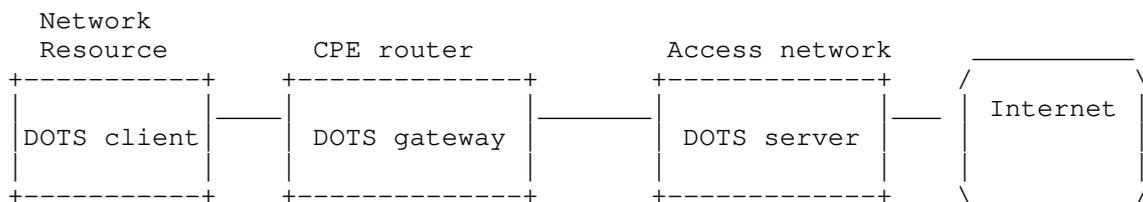
Figure 1: Sample DOTS Deployment (1)

The DOTS server can also be running on the Internet, as depicted in
Figure 2.
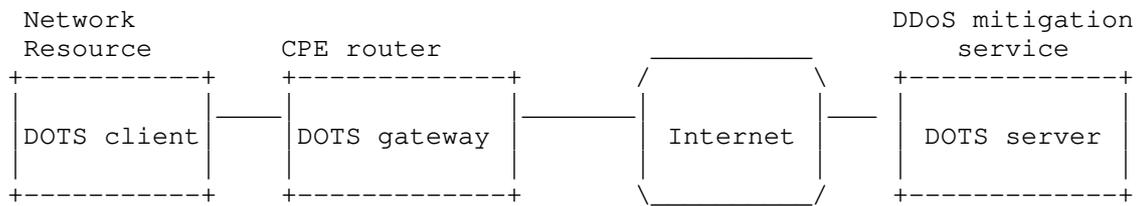
```
   Network                                                  DDoS mitigation
   Resource              CPE router            _____         service
+----------+       +-------------+           /          \   +-------------+
|          |       |             |          |            |   |             |
|DOTS client|_____|DOTS gateway |_____| Internet   |___|  DOTS server|
|          |       |             |          |            |   |             |
|          |       |             |          |            |   |             |
+----------+       +-------------+           _____/    +-------------+
```

                   Figure 2: Sample DOTS Deployment (2)

   In typical deployments, the DOTS client belongs to a different
   administrative domain than the DOTS server. For example, the DOTS
   client is a firewall protecting services owned and operated by an
   domain, while the DOTS server is owned and operated by a different
   domain providing DDoS mitigation services. That domain providing
   DDoS mitigation service might, or might not, also provide Internet
   access service to the website operator.

   The DOTS server may (not) be co-located with the DOTS mitigator. In
   typical deployments, the DOTS server belongs to the same
   administrative domain as the mitigator.

   The DOTS client can communicate directly with the DOTS server or
   indirectly via a DOTS gateway.

   This document focuses on the DOTS signal channel.

4. Happy Eyeballs for DOTS Signal Channel

   DOTS signaling can happen with DTLS [RFC6347] over UDP and TLS
   [RFC5246] over TCP. A DOTS client can use DNS to determine the IP
   address(es) of a DOTS server or a DOTS client may be provided with
   the list of DOTS server IP addresses. The DOTS client MUST know a
   DOTS server's domain name; hard-coding the domain name of the DOTS
   server into software is NOT RECOMMENDED in case the domain name is
   not valid or needs to change for legal or other reasons. The DOTS
   client performs A and/or AAAA record lookup of the domain name and
   the result will be a list of IP addresses, each of which can be used
   to contact the DOTS server using UDP and TCP.

   If an IPv4 path to reach a DOTS server is found, but the DOTS
   server's IPv6 path is not working, a dual-stack DOTS client can
   experience a significant connection delay compared to an IPv4-only
   DOTS client. The other problem is that if a middlebox between the
   DOTS client and DOTS server is configured to block UDP, the DOTS
   client will fail to establish a DTLS session with the DOTS server and
   will, then, have to fall back to TLS over TCP incurring significant
   connection delays. [I-D.ietf-dots-requirements] discusses that DOTS

```

client and server will have to support both connectionless and
connection-oriented protocols.

To overcome these connection setup problems, the DOTS client can try
connecting to the DOTS server using both IPv6 and IPv4, and try both
DTLS over UDP and TLS over TCP in a fashion similar to the Happy
Eyeballs mechanism [RFC6555].  These connection attempts are
performed by the DOTS client when its initializes, and the client
uses that information for its subsequent alert to the DOTS server.
In order of preference (most preferred first), it is UDP over IPv6,
UDP over IPv4, TCP over IPv6, and finally TCP over IPv4, which
adheres to address preference order [RFC6724] and the DOTS preference
that UDP be used over TCP (to avoid TCP's head of line blocking).

```
DOTS client   DOTS server
    | |
    |--DTLS ClientHello, IPv6 ---->X |
    |--TCP SYN, IPv6-------------->X |
    |--DTLS ClientHello, IPv4 ---->X |
    |--TCP SYN, IPv4--->|
    |--DTLS ClientHello, IPv6 ---->X |
    |--TCP SYN, IPv6-------------->X |
    |<-TCP SYNACK--|
    |--DTLS ClientHello, IPv4 ---->X |
    |--TCP ACK--->|
    |<-----------Establish TLS Session-------------------------->|
    |---------------DOTS signal--------------------------------->|
    | |
```

Figure 3: Happy Eyeballs

In reference to Figure 3, the DOTS client sends two TCP SYNs and two
DTLS ClientHello messages at the same time over IPv6 and IPv4.  In
this example, it is assumed that the IPv6 path is broken and UDP is
dropped by a middlebox but has little impact to the DOTS client
because there is no long delay before using IPv4 and TCP.  The DOTS
client repeats the mechanism to discover if DOTS signaling with DTLS
over UDP becomes available from the DOTS server, so the DOTS client
can migrate the DOTS signal channel from TCP to UDP, but such probing
SHOULD NOT be done more frequently than every 24 hours and MUST NOT
be done more frequently than every 5 minutes.

5.  DOTS Signal Channel

5.1.  Overview

   The DOTS signal channel is built on top of the Constrained
   Application Protocol (CoAP) [RFC7252], a lightweight protocol
   originally designed for constrained devices and networks.  CoAP's
   expectation of packet loss, support for asynchronous non-confirmable
   messaging, congestion control, small message overhead limiting the
   need for fragmentation, use of minimal resources, and support for
   (D)TLS make it a good foundation on which to build the DOTS signaling
   mechanism.

   The DOTS signal channel is layered on existing standards (Figure 4).

   By default, DOTS signal channel MUST run over port number TBD as
   defined in Section 10.1, for both UDP and TCP, unless the DOTS server
   has mutual agreement with its DOTS clients to use a port other than
   TBD for DOTS signal channel, or DOTS clients supports means to
   dynamically discover the ports used by their DOTS servers.  In order
   to use a distinct port number (vs.  TBD), DOTS clients and servers
   should support a configurable parameter to supply the port number to
   use.

```
                          +--------------+
                          |     DOTS     |
                          +--------------+
                          |     CoAP     |
                          +--------------+
                          | TLS | DTLS   |
                          +--------------+
                          | TCP |  UDP   |
                          +--------------+
                          |      IP      |
                          +--------------+
```

           Figure 4: Abstract Layering of DOTS signal channel over CoAP over
                                  (D)TLS

   The signal channel is initiated by the DOTS client. Once the signal
   channel is established, the DOTS agents periodically send heartbeats
   to keep the channel active. At any time, the DOTS client may send a
   mitigation request message to the DOTS server over the active
   channel. While mitigation is active, due to the higher likelihood of
   packet loss during a DDoS attack, the DOTS server periodically sends
   status messages to the client, including basic mitigation feedback
   details. Mitigation remains active until the DOTS client explicitly
   terminates mitigation, or the mitigation lifetime expires.

Messages exchanged between DOTS client and server are serialized
using Concise Binary Object Representation (CBOR) [RFC7049], CBOR is
a binary encoding designed for small code and message size. CBOR
encoded payloads are used to convey signal channel specific payload
messages that convey request parameters and response information such
as errors. This specification uses the encoding rules defined in
[I-D.ietf-core-yang-cbor] for representing mitigation scope and DOTS
signal channel session configuration data defined using YANG
(Section 5.2) as CBOR data.

DOTS agents MUST support GET, PUT, and DELETE CoAP methods. The
payload included in CoAP responses with 2.xx and 3.xx Response Codes
MUST be of content type "application/cbor" (Section 5.5.1 of
[RFC7252]). CoAP responses with 4.xx and 5.xx error Response Codes
MUST include a diagnostic payload (Section 5.5.2 of [RFC7252]). The
Diagnostic Payload may contain additional information to aid
troubleshooting.

5.2. DOTS Signal YANG Module

This document defines a YANG [RFC6020] module for mitigation scope
and DOTS signal channel session configuration data.

5.2.1. Mitigation Request YANG Module Tree Structure

This document defines the YANG module "ietf-dots-signal", which has
the following tree structure:

```
module: ietf-dots-signal
   +--rw mitigation-scope
      +--rw client-identifier*   binary
      +--rw scope* [mitigation-id]
         +--rw mitigation-id        int32
         +--rw target-ip*           inet:ip-address
         +--rw target-prefix*       inet:ip-prefix
         +--rw target-port-range* [lower-port upper-port]
         |  +--rw lower-port    inet:port-number
         |  +--rw upper-port    inet:port-number
         +--rw target-protocol*     uint8
         +--rw fqdn*                inet:domain-name
         +--rw uri*                 inet:uri
         +--rw alias-name*          string
         +--rw lifetime?            int32
```

5.2.2. Mitigation Request YANG Module

<CODE BEGINS> file "ietf-dots-signal@2017-10-04.yang"

```
module ietf-dots-signal {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal";
     prefix "signal";

     import ietf-inet-types {
         prefix "inet";
     }

  organization "IETF DOTS Working Group";

  contact
    "Konda, Tirumaleswar Reddy <TirumaleswarReddy_Konda@McAfee.com>
     Mohamed Boucadair <mohamed.boucadair@orange.com>
     Prashanth Patil <praspati@cisco.com>
     Andrew Mortensen <amortensen@arbor.net>
     Nik Teague <nteague@verisign.com>";

   description
     "This module contains YANG definition for DOTS
     signal sent by the DOTS client to the DOTS server.

      Copyright (c) 2017 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

   revision 2017-10-04 {
     description
         "Add units and fix some nits.";
     reference
     "-05";
   }

   revision 2017-08-03 {
     reference
```

```
      "https://tools.ietf.org/html/draft-reddy-dots-signal-channel";
   }

container mitigation-scope {
    description
      "Top level container for a mitigation request.";

    leaf-list client-identifier {
       type binary;
       description
         "A client identifier conveyed by a DOTS gateway
          to a remote DOTS server.";
    }

    list scope {
       key mitigation-id;
       description "Identifier for the mitigation request.";

       leaf mitigation-id {
         type int32;
         description "Mitigation request identifier.";
        }
        leaf-list target-ip {
           type inet:ip-address;
           description
              "IPv4 or IPv6 address identifying the target.";
        }

        leaf-list target-prefix {
           type inet:ip-prefix;
           description
              "IPv4 or IPv6 prefix identifying the target.";
        }

        list target-port-range {
           key "lower-port upper-port";

           description "Port range. When only lower-port is present,
                        it represents a single port.";

           leaf lower-port {
              type inet:port-number;
              mandatory true;
              description "Lower port number.";
           }

           leaf upper-port {
              type inet:port-number;
```

```
                            must ". >= ../lower-port" {
                              error-message
                              "The upper port number must be greater than or
                               equal to lower port number.";
                            }
                            description "Upper port number.";
                    }
              }

              leaf-list target-protocol {
                  type uint8;
                  description "Identifies the target protocol number.";
              }

              leaf-list fqdn {
                   type inet:domain-name;
                   description "FQDN";
              }

              leaf-list uri {
                  type inet:uri;
                  description "URI";
              }

              leaf-list alias-name {
                  type string;
                  description "alias name";
              }

              leaf lifetime {
                  type int32;
                  units "seconds";
                  default 3600;

                  description
                    "Indicates the lifetime of the mitigation request.";
              }
          }
      }
   }
<CODE ENDS>
```

5.2.3. Session Configuration YANG Module Tree Structure

   This document defines the YANG module "ietf-dots-signal-config",
   which has the following structure:

```
module: ietf-dots-signal-config
   +--rw signal-config
      +--rw session-id?          int32
      +--rw heartbeat-interval?  int16
      +--rw missing-hb-allowed?  int16
      +--rw max-retransmit?      int16
      +--rw ack-timeout?         int16
      +--rw ack-random-factor?   decimal64
      +--rw trigger-mitigation?  boolean
```

5.2.4. Session Configuration YANG Module

<CODE BEGINS> file "ietf-dots-signal-config@2017-10-04.yang"

```
module ietf-dots-signal-config {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal-config";
    prefix "config";

   organization "IETF DOTS Working Group";

   contact
     "Konda, Tirumaleswar Reddy <TirumaleswarReddy_Konda@McAfee.com>
      Mohamed Boucadair <mohamed.boucadair@orange.com>
      Prashanth Patil <praspati@cisco.com>
      Andrew Mortensen <amortensen@arbor.net>
      Nik Teague <nteague@verisign.com>";

    description
      "This module contains YANG definition for DOTS
      signal channel session configuration.

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2017-10-04 {
      description
          "Add units/defaults and fix some nits.";
```

```
        reference
        "-05";
      }

      revision 2016-11-28 {
        reference
        "https://tools.ietf.org/html/draft-reddy-dots-signal-channel";
      }

      container signal-config {
          description "Top level container for DOTS signal channel session
                      configuration.";

          leaf session-id {
              type int32;
              description "An identifier for the DOTS signal channel
                          session configuration data.";
          }

          leaf heartbeat-interval {
              type int16;
              units "seconds";
              default 30;

              description
                "DOTS agents regularly send heartbeats to each other
                 after mutual authentication in order to keep
                 the DOTS signal channel open.";
          }

          leaf missing-hb-allowed {
              type int16;
              default 5;

              description
                "Maximum number of missing heartbeats allowed.";
          }

          leaf max-retransmit {
              type int16;
              default 3;

              description
                "Maximum number of retransmissions of a
                 Confirmable message.";
          }

          leaf ack-timeout {
```

```
            type int16;
            units "seconds";
            default 2;

            description
               "Initial retransmission timeout value.";
         }

         leaf ack-random-factor {
            type decimal64 {
            fraction-digits 2;
             }

            default 1.5;

             description
                   "Random factor used to influence the timing of
                    retransmissions";
         }
         leaf trigger-mitigation {
            type boolean;
            default true;

            description
               "If false, then mitigation is triggered
                only when the DOTS server channel session is lost";
         }
      }
}
<CODE ENDS>
```

5.3. CoAP URIs

   The DOTS server MUST support the use of the path-prefix of "/.well-
   known/" as defined in [RFC5785] and the registered name of "dots".
   Each DOTS operation is indicated by a path-suffix that indicates the
   intended operation.

```
+------------------------+----------------+-------------------+
| Operation              |Operation path  | Details           |
+========================+================+===================+
| Mitigation             | /v1/mitigate   | Section 5.4       |
|                        |                |                   |
+------------------------+----------------+-------------------+
| Session configuration  | /v1/config     | Section 5.5       |
|                        |                |                   |
+------------------------+----------------+-------------------+
```

              Figure 5: Operations and their corresponding URIs:

5.4. Mitigation Request

   The following methods are used to request or withdraw mitigation
   requests:

   PUT: DOTS clients use the PUT method to request mitigation
      (Section 5.4.1). During active mitigation, DOTS clients may use
      PUT requests to convey mitigation efficacy updates to the DOTS
      server (Section 5.4.4).

   DELETE: DOTS clients use the DELETE method to withdraw a request for
      mitigation from the DOTS server (Section 5.4.2).

   GET: DOTS clients may use the GET method to subscribe to DOTS server
      status messages, or to retrieve the list of existing mitigations
      (Section 5.4.3).

   Mitigation request and response messages are marked as Non-
   confirmable messages. DOTS agents SHOULD follow the data
   transmission guidelines discussed in Section 3.1.3 of [RFC8085] and
   control transmission behavior by not sending on average more than one
   UDP datagram per RTT to the peer DOTS agent.

   Requests marked by the DOTS client as Non-confirmable messages are
   sent at regular intervals until a response is received from the DOTS
   server and if the DOTS client cannot maintain a RTT estimate then it
   SHOULD NOT send more than one Non-confirmable request every 3
   seconds, and SHOULD use an even less aggressive rate when possible
   (case 2 in Section 3.1.3 of [RFC8085]).

5.4.1. Requesting mitigation

   When a DOTS client requires mitigation for any reason, the DOTS
   client uses CoAP PUT method to send a mitigation request to the DOTS
   server (Figure 6, illustrated in JSON diagnostic notation). The DOTS
   server can enable mitigation on behalf of the DOTS client by

communicating the DOTS client's request to the mitigator and relaying
selected mitigator feedback to the requesting DOTS client.

```
Header: PUT (Code=0.03)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "version"
Uri-Path: "mitigate"
Content-Type: "application/cbor"
{
  "mitigation-scope": {
    "client-identifier": [
      "string"
    ],
    "scope": [
      {
        "mitigation-id": integer,
        "target-ip": [
          "string"
         ],
        "target-prefix": [
          "string"
         ],
        "target-port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
         ],
        "target-protocol": [
          integer
         ],
        "fqdn": [
          "string"
        ],
        "uri": [
          "string"
        ],
        "alias-name": [
          "string"
        ],
        "lifetime": integer
      }
    ]
  }
}
```

                    Figure 6: PUT to convey DOTS signals

   The parameters are described below.

client-identifier: The client identifier MAY be conveyed by the DOTS
   gateway to propagate the DOTS client identity from the gateway's
   client-side to the gateway's server-side, and from the gateway's
   server-side to the DOTS server. This allows the final DOTS server
   to accept mitigation requests with scopes which the DOTS client is
   authorized to manage.

   The 'client-identifier' value MUST be assigned by the DOTS gateway
   in a manner that ensures that there is no probability that the
   same value will be assigned to a different DOTS client. The DOTS
   gateway MUST obscure potentially sensitive DOTS client identity
   information. The client-identifier attribute SHOULD NOT to be
   generated and included by the DOTS client.

   This is an optional attribute.

mitigation-id: Identifier for the mitigation request represented
   using an integer. This identifier MUST be unique for each
   mitigation request bound to the DOTS client, i.e., the mitigation-
   id parameter value in the mitigation request needs to be unique
   relative to the mitigation-id parameter values of active
   mitigation requests conveyed from the DOTS client to the DOTS
   server. This identifier MUST be generated by the DOTS client.
   This document does not make any assumption about how this
   identifier is generated. This is a mandatory attribute.

target-ip: A list of IP addresses under attack. This is an optional
   attribute.

target-prefix: A list of prefixes under attack. Prefixes are
   represented using CIDR notation [RFC4632]. This is an optional
   attribute.

target-port-range: A list of ports under attack. The port range,
   lower-port for lower port number and upper-port for upper port
   number. When only lower-port is present, it represents a single
   port. For TCP, UDP, Stream Control Transmission Protocol (SCTP)
   [RFC4960], or Datagram Congestion Control Protocol (DCCP)
   [RFC4340]: the range of ports (e.g., 1024-65535). This is an
   optional attribute.

target-protocol: A list of protocols under attack. Values are taken
   from the IANA protocol registry [proto_numbers]. The value 0 has
   a special meaning for 'all protocols'. This is an optional
   attribute.

fqdn: A list of Fully Qualified Domain Names. Fully Qualified
   Domain Name (FQDN) is the full name of a system, rather than just

its hostname. For example, "venera" is a hostname, and
"venera.isi.edu" is an FQDN. This is an optional attribute.

uri: A list of Uniform Resource Identifiers (URI). This is an
optional attribute.

alias-name: A list of aliases. Aliases can be created using the
DOTS data channel (Section 3.1.1 of [I-D.ietf-dots-data-channel])
or direct configuration, or other means and then used in
subsequent signal channel exchanges to refer more efficiently to
the resources under attack. This is an optional attribute.

lifetime: Lifetime of the mitigation request in seconds. The
default lifetime of a mitigation request is 3600 seconds (60
minutes) -- this value was chosen to be long enough so that
refreshing is not typically a burden on the DOTS client, while
expiring the request where the client has unexpectedly quit in a
timely manner.

A lifetime of negative one (-1) indicates indefinite lifetime for
the mitigation request.

DOTS clients SHOULD include this parameter in their mitigation
requests. If no lifetime is supplied by a DOTS client, the DOTS
server uses the default lifetime value (3600 seconds). Upon the
expiry of this lifetime, and if the request is not refreshed, the
mitigation request is removed. The request can be refreshed by
sending the same request again. The server MAY refuse indefinite
lifetime, for policy reasons; the granted lifetime value is
returned in the response. DOTS clients MUST be prepared to not be
granted mitigations with indefinite lifetimes. The server MUST
always indicate the actual lifetime in the response and the
remaining lifetime in status messages sent to the client. This is
a mandatory parameter for responses.

The CBOR key values for the parameters are defined in Section 6.
Section 10 defines how the CBOR key values can be allocated to
standards bodies and vendors.

FQDN and URI mitigation scopes may be thought of as a form of scope
alias, in which the addresses to which the domain name or URI resolve
represent the full scope of the mitigation.

In the PUT request at least one of the attributes 'target-ip' or
'target-prefix' or 'fqdn' or 'uri 'or 'alias-name' MUST be present.
DOTS agents can safely ignore Vendor-Specific parameters they don't
understand.

The relative order of two mitigation requests from a DOTS client is determined by comparing their respective 'mitigation-id' values. If two mitigation requests have overlapping mitigation scopes, the mitigation request with higher numeric 'mitigation-id' value will override the mitigation request with a lower numeric 'mitigation-id' value. Two mitigation-ids are overlapping if there is a common IP address, IP prefix, FQDN, URI, or alias-name. The overlapped lower numeric 'mitigation-id' MUST be automatically deleted and no longer available at the DOTS server.

The Uri-Path option carries a major and minor version nomenclature to manage versioning and DOTS signal channel in this specification uses v1 major version.

If the DOTS client is using the certificate provisioned by the Enrollment over Secure Transport (EST) server [RFC6234] in the DOTS gateway-domain to authenticate itself to the DOTS gateway, then the 'client-identifier' value can be the output of a cryptographic hash algorithm whose input is the DER-encoded ASN.1 representation of the Subject Public Key Info (SPKI) of an X.509 certificate. In this version of the specification, the cryptographic hash algorithm used is SHA-256 [RFC6234]. The output of the cryptographic hash algorithm is truncated to 16 bytes; truncation is done by stripping off the final 16 bytes. The truncated output is base64url encoded. If the 'client-identifier' value is already present in the mitigation request received from the DOTS client, the DOTS gateway MAY compute the 'client-identifier' value, as discussed above, and add the computed 'client-identifier' value to the end of the 'client-identifier' list. The DOTS server MUST NOT use the 'client-identifier' for the DOTS client authentication process.

In both DOTS signal and data channel sessions, the DOTS client MUST authenticate itself to the DOTS server (Section 9). The DOTS server may use the algorithm in Section 7 of [RFC7589] to derive the DOTS client identity or username from the client certificate. The DOTS client identity allows the DOTS server to accept mitigation requests with scopes which the DOTS client is authorized to manage. The DOTS server couples the DOTS signal and data channel sessions using the DOTS client identity and the 'client-identifier' parameter value, so the DOTS server can validate whether the aliases conveyed in the mitigation request were indeed created by the same DOTS client using the DOTS data channel session. If the aliases were not created by the DOTS client, the DOTS server returns 4.00 (Bad Request) in the response.

The DOTS server couples the DOTS signal channel sessions using the DOTS client identity and the 'client-identifier' parameter value, and the DOTS server uses 'mitigation-id' parameter value to detect

   duplicate mitigation requests. If the mitigation request contains
   both alias-name and other parameters identifying the target resources
   (such as, 'target-ip', 'target-prefix', 'target-port-range', 'fqdn',
   or 'uri'), then the DOTS server appends the parameter values in
   'alias-name' with the corresponding parameter values in 'target-ip',
   'target-prefix', 'target-port-range', 'fqdn', or 'uri'.

   Figure 7 shows a PUT request example to signal that ports 80, 8080,
   and 443 on the servers 2001:db8:6401::1 and 2001:db8:6401::2 are
   being attacked (illustrated in JSON diagnostic notation).

```
Header: PUT (Code=0.03)
Uri-Host: "www.example.com"
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "v1"
Uri-Path: "mitigate"
Content-Format: "application/cbor"
{
  "mitigation-scope": {
    "client-identifier": [
      "dz6pHjaADkaFTbjr0JGBpw"
    ],
    "scope": [
      {
        "mitigation-id": 12332,
        "target-ip": [
          "2001:db8:6401::1",
          "2001:db8:6401::2"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          },
          {
            "lower-port": 443
          },
          {
            "lower-port": 8080
          }
        ],
        "target-protocol": [
          6
        ]
      }
    ]
  }
}
```

The CBOR encoding format is shown below:

```
A1                                       # map(1)
   01                                    # unsigned(1)
   A2                                    # map(2)
      18 20                              # unsigned(32)
      81                                 # array(1)
      76                                 # text(22)
         647A3670486A6141446B614654626A72304A47427077 # "dz6pHjaADkaFTbjr0JGBpw"
      02                                 # unsigned(2)
      81                                 # array(1)
         A4                              # map(4)
            03                           # unsigned(3)
            19 302C                      # unsigned(12332)
            04                           # unsigned(4)
            82                           # array(2)
               70                        # text(16)
                  323030313A6462383A363430313A3A31 # "2001:db8:6401::1"
               70                        # text(16)
                  323030313A6462383A363430313A3A32 # "2001:db8:6401::2"
            05                           # unsigned(5)
            83                           # array(3)
               A1                        # map(1)
                  06                     # unsigned(6)
                  18 50                  # unsigned(80)
               A1                        # map(1)
                  06                     # unsigned(6)
                  19 01BB                # unsigned(443)
               A1                        # map(1)
                  06                     # unsigned(6)
                  19 1F90                # unsigned(8080)
            08                           # unsigned(8)
            81                           # array(1)
               06                        # unsigned(6)
```

Figure 7: PUT for DOTS signal

The DOTS server indicates the result of processing the PUT request
using CoAP response codes. CoAP 2.xx codes are success. CoAP 4.xx
codes are some sort of invalid requests. Figure 8 shows a PUT
response for CoAP 2.xx response codes.

```
{
   "mitigation-scope": {
      "client-identifier": [
         "string"
      ],
      "scope": [
         {
            "mitigation-id": integer,
            "lifetime": integer
         }
      ]
   }
}
```

                     Figure 8: 2.xx response body

   COAP 5.xx codes are returned if the DOTS server has erred or is
   currently unavailable to provide mitigation in response to the
   mitigation request from the DOTS client.

   If the DOTS server does not find the 'mitigation-id' parameter value
   conveyed in the PUT request in its configuration data, then the
   server MAY accept the mitigation request by sending back a 2.01
   (Created) response to the DOTS client; the DOTS server will
   consequently try to mitigate the attack.

   If the DOTS server finds the 'mitigation-id' parameter value conveyed
   in the PUT request in its configuration data, then the server MAY
   update the mitigation request, and a 2.04 (Changed) response is
   returned to indicate a successful update of the mitigation request.

   If the request is missing one or more mandatory attributes, then 4.00
   (Bad Request) will be returned in the response or if the request
   contains invalid or unknown parameters then 4.02 (Invalid query) is
   returned in the response.

   For a mitigation request to continue beyond the initial negotiated
   lifetime, the DOTS client need to refresh the current mitigation
   request by sending a new PUT request. The PUT request MUST use the
   same 'mitigation-id' value, and MUST repeat all the other parameters
   as sent in the original mitigation request apart from a possible
   change to the lifetime parameter value.

   A DOTS gateway MUST update the 'client-identifier' list in the
   response to remove the 'client-identifier' value it had added in the
   corresponding request before forwarding the response to the DOTS
   client.

5.4.2. Withdraw a DOTS Signal

   A DELETE request is used to withdraw a DOTS signal from a DOTS server
   (Figure 9).

     Header: DELETE (Code=0.04)
     Uri-Host: "host"
     Uri-Path: ".well-known"
     Uri-Path: "dots"
     Uri-Path: "version"
     Uri-Path: "mitigate"
     Content-Format: "application/cbor"
     {
       "mitigation-scope": {
         "client-identifier": [
           "string"
         ],
         "scope": [
           {
             "mitigation-id": integer
           }
         ]
       }
     }

                       Figure 9: Withdraw DOTS signal

   The DOTS server immediately acknowledges a DOTS client's request to
   withdraw the DOTS signal using 2.02 (Deleted) response code with no
   response payload. A 2.02 (Deleted) Response Code is returned even if
   the 'mitigation-id' parameter value conveyed in the DELETE request
   does not exist in its configuration data before the request.

   If the DOTS server finds the 'mitigation-id' parameter value conveyed
   in the DELETE request in its configuration data, then to protect
   against route or DNS flapping caused by a client rapidly toggling
   mitigation, and to dampen the effect of oscillating attacks, DOTS
   servers MAY allow mitigation to continue for a limited period after
   acknowledging a DOTS client's withdrawal of a mitigation request.
   During this period, the DOTS server status messages SHOULD indicate
   that mitigation is active but terminating. The initial active-but-
   terminating period SHOULD be sufficiently long to absorb latency
   incurred by route propagation. The active-but-terminating period
   SHOULD be set by default to 120 seconds. If the client requests
   mitigation again before the initial active-but-terminating period
   elapses, the DOTS server MAY exponentially increase the active-but-
   terminating period up to a maximum of 300 seconds (5 minutes). After
   the active-but-terminating period elapses, the DOTS server MUST treat

the mitigation as terminated, as the DOTS client is no longer
responsible for the mitigation. For example, if there is a financial
relationship between the DOTS client and server domains, the DOTS
client ceases incurring cost at this point.

5.4.3. Retrieving a DOTS Signal

A GET request is used to retrieve information (including status) of a
DOTS signal from a DOTS server (Figure 10). If the DOTS server does
not find the 'mitigation-id' parameter value conveyed in the GET
request in its configuration data, then it responds with a 4.04 (Not
Found) error response code. The 'c' (content) parameter and its
permitted values defined in [I-D.ietf-core-comi] can be used to
retrieve non-configuration data (attack mitigation status) or
configuration data or both. The DOTS server SHOULD support this
optional filtering capability but can safely ignore it if not
supported.

The examples below assume the default of "c=a".

   1) To retrieve all DOTS signals signaled by the DOTS client.

      Header: GET (Code=0.01)
      Uri-Host: "host"
      Uri-Path: ".well-known"
      Uri-Path: "dots"
      Uri-Path: "version"
      Uri-Path: "mitigate"
      Observe : 0
      {
        "mitigation-scope": {
          "client-identifier": [
            "string"
          ]
        }
      }

   2) To retrieve a specific DOTS signal signaled by the DOTS client.
      The configuration data in the response will be formatted in the
      same order it was processed at the DOTS server.

      Header: GET (Code=0.01)
      Uri-Host: "host"
      Uri-Path: ".well-known"
      Uri-Path: "dots"
      Uri-Path: "version"
      Uri-Path: "mitigate"
      Observe : 0
      Content-Format: "application/cbor"
      {
        "mitigation-scope": {
          "client-identifier": [
            "string"
          ],
          "scope": [
            {
              "mitigation-id": integer
            }
          ]
        }
      }

                    Figure 10: GET to retrieve the rules

   Figure 11 shows a response example of all the active mitigation
   requests associated with the DOTS client on the DOTS server and the
   mitigation status of each mitigation request.

```
   {
     "mitigation-scope": {
        "scope": [
           {
              "mitigation-id": 12332,
              "mitigation-start": 1507818434.00,
              "target-protocol": [
                   17
               ],
              "lifetime":1800,
              "status":2,
              "bytes-dropped": 134334555,
              "bps-dropped":  43344,
              "pkts-dropped": 333334444,
              "pps-dropped": 432432
            },
           {
              "mitigation-id": 12333,
              "mitigation-start": 1507818393.00,
              "target-protocol": [
                   6
               ],
              "lifetime":1800,
              "status":3
              "bytes-dropped": 0,
              "bps-dropped":  0,
              "pkts-dropped": 0,
              "pps-dropped": 0
           }
        ]
     }
   }
```

                      Figure 11: Response body

   The mitigation status parameters are described below.

   lifetime: The remaining lifetime of the mitigation request in
      seconds.

   mitigation-start: Mitigation start time is represented in seconds
      relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of
      [RFC7049]). The encoding is modified so that the leading tag 1
      (epoch-based date/time) MUST be omitted.

   bytes-dropped: The total dropped byte count for the mitigation
      request since the attack mitigation is triggered. The count wraps

around when it reaches the maximum value of unsigned integer.
This is an optional attribute.

bps-dropped: The average dropped bytes per second for the mitigation
request since the attack mitigation is triggered. This SHOULD be
a five-minute average. This is an optional attribute.

pkts-dropped: The total dropped packet count for the mitigation
request since the attack mitigation is triggered. This is an
optional attribute.

pps-dropped: The average dropped packets per second for the
mitigation request since the attack mitigation is triggered. This
SHOULD be a five-minute average. This is an optional attribute.

status: Status of attack mitigation. The 'status' parameter is a
mandatory attribute.

The various possible values of 'status' parameter are explained
below:

Parameter value	Description
1	Attack mitigation is in progress (e.g., changing the network path to re-route the inbound traffic to DOTS mitigator).
2	Attack is successfully mitigated (e.g., traffic is redirected to a DDOS mitigator and attack traffic is dropped).
3	Attack has stopped and the DOTS client can withdraw the mitigation request.
4	Attack has exceeded the mitigation provider capability.
5	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.

The observe option defined in [RFC7641] extends the CoAP core
protocol with a mechanism for a CoAP client to "observe" a resource
on a CoAP server: the client retrieves a representation of the
resource and requests this representation be updated by the server as
long as the client is interested in the resource. A DOTS client

conveys the observe option set to 0 in the GET request to receive
unsolicited notifications of attack mitigation status from the DOTS
server. Unidirectional notifications within the bidirectional signal
channel allows unsolicited message delivery, enabling asynchronous
notifications between the agents. Due to the higher likelihood of
packet loss during a DDoS attack, DOTS server periodically sends
attack mitigation status to the DOTS client and also notifies the
DOTS client whenever the status of the attack mitigation changes. If
the DOTS server cannot maintain a RTT estimate then it SHOULD NOT
send more than one unsolicited notification every 3 seconds, and
SHOULD use an even less aggressive rate when possible (case 2 in
Section 3.1.3 of [RFC8085]). A DOTS client that is no longer
interested in receiving notifications from the DOTS server can simply
"forget" the observation. When the DOTS server then sends the next
notification, the DOTS client will not recognize the token in the
message and thus will return a Reset message. This causes the DOTS
server to remove the associated entry. Alternatively, the DOTS
client can explicitly deregister by issuing a GET request that has
the Token field set to the token of the observation to be cancelled
and includes an Observe Option with the value set to 1 (deregister).

```
         DOTS Client                 DOTS Server
            |                             |
            |  GET /<mitigation-id number>|
            |  Token: 0x4a                |      Registration
            |  Observe: 0                 |
            +---------------------------->|
            |                             |
            |  2.05 Content               |
            |  Token: 0x4a                |      Notification of
            |  Observe: 12                |      the current state
            |  status: "mitigation        |
            |          in progress"       |
            |<----------------------------+
            |  2.05 Content               |
            |  Token: 0x4a                |      Notification upon
            |  Observe: 44                |       a state change
            |  status: "mitigation        |
            |          complete"          |
            |<----------------------------+
            |  2.05 Content               |
            |  Token: 0x4a                |      Notification upon
            |  Observe: 60                |      a state change
            |  status: "attack stopped"   |
            |<----------------------------+
            |                             |
```

Figure 12: Notifications of attack mitigation status

5.4.3.1. Mitigation Status

   The DOTS client can send the GET request at frequent intervals
   without the Observe option to retrieve the configuration data of the
   mitigation request and non-configuration data (i.e., the attack
   status). The frequency of polling the DOTS server to get the
   mitigation status should follow the transmission guidelines given in
   Section 3.1.3 of [RFC8085]. If the DOTS server has been able to
   mitigate the attack and the attack has stopped, the DOTS server
   indicates as such in the status, and the DOTS client recalls the
   mitigation request by issuing a DELETE for the mitigation-id.

   A DOTS client should react to the status of the attack from the DOTS
   server and not the fact that it has recognized, using its own means,
   that the attack has been mitigated. This ensures that the DOTS
   client does not recall a mitigation request in a premature fashion
   because it is possible that the DOTS client does not sense the DDOS
   attack on its resources but the DOTS server could be actively
   mitigating the attack and the attack is not completely averted.

5.4.4. Efficacy Update from DOTS Client

   While DDoS mitigation is active, due to the likelihood of packet
   loss, a DOTS client MAY periodically transmit DOTS mitigation
   efficacy updates to the relevant DOTS server. A PUT request
   (Figure 13) is used to convey the mitigation efficacy update to the
   DOTS server.

   The PUT request MUST include all the parameters used in the PUT
   request to convey the DOTS signal (Section 5.4.1) unchanged apart
   from the lifetime parameter value. If this is not the case, the DOTS
   server MUST reject the request with a 4.02 error response code.

   The If-Match Option (Section 5.10.8.1 of [RFC7252]) with an empty
   value is used to make the PUT request conditional on the current
   existence of the mitigation request. If UDP is used as transport,
   CoAP requests may arrive out-of-order. For example, the DOTS client
   may send a PUT request to convey an efficacy update to the DOTS
   server followed by a DELETE request to withdraw the mitigation
   request, but the DELETE request arrives at the DOTS server before the
   PUT request. To handle out-of-order delivery of requests, if an If-
   Match option is present in the PUT request and the 'mitigation-id' in
   the request matches a mitigation request from that DOTS client, then
   the request is processed. If no match is found, the PUT request is
   silently ignored.

```
    Header: PUT (Code=0.03)
    Uri-Host: "host"
    Uri-Path: ".well-known"
    Uri-Path: "dots"
    Uri-Path: "version"
    Uri-Path: "mitigate"
    Content-Format: "application/cbor"
    {
     "mitigation-scope": {
       "client-identifier": [
          "string"
       ],
       "scope": [
         {
           "mitigation-id": integer,
           "target-ip": [
             "string"
            ],
           "target-port-range": [
              {
                "lower-port": integer,
                "upper-port": integer
              }
            ],
            "target-protocol": [
             integer
            ],
            "fqdn": [
             "string"
            ],
            "uri": [
             "string"
            ],
            "alias-name": [
             "string"
            ],
          "lifetime": integer,
          "attack-status": integer
        }
      ]
    }
  }
```

                       Figure 13: Efficacy Update

   The 'attack-status' parameter is a mandatory attribute when doing a
   efficacy update. The various possible values contained in the
   'attack-status' parameter are described below:

```
/-------------------+----------------------------------------------------\
| Parameter value   | Description                                        |
+-------------------+----------------------------------------------------+
| 1                 | DOTS client determines that it is still under attack.|
+-------------------+----------------------------------------------------+
| 2                 | DOTS client determines that the attack is          |
|                   | successfully mitigated                             |
|                   | (e.g., attack traffic is not seen).                |
\-------------------+----------------------------------------------------/
```

The DOTS server indicates the result of processing a PUT request
using CoAP response codes. The response code 2.04 (Changed) is
returned if the DOTS server has accepted the mitigation efficacy
update. The error response code 5.03 (Service Unavailable) is
returned if the DOTS server has erred or is incapable of performing
the mitigation.

5.5. DOTS Signal Channel Session Configuration

The DOTS client can negotiate, configure, and retrieve the DOTS
signal channel session behavior. The DOTS signal channel can be
used, for example, to configure the following:

a. Heartbeat interval: DOTS agents regularly send heartbeats (CoAP
    Ping/Pong) to each other after mutual authentication in order to
    keep the DOTS signal channel open, heartbeat messages are
    exchanged between the DOTS agents every heartbeat-interval
    seconds to detect the current status of the DOTS signal channel
    session.

b. Missing heartbeats allowed: This variable indicates the maximum
    number of consecutive heartbeat messages for which a DOTS agent
    did not receive a response before concluding that the session is
    disconnected or defunct.

c. Acceptable signal loss ratio: Maximum retransmissions,
    retransmission timeout value and other message transmission
    parameters for the DOTS signal channel.

Reliability is provided to requests and responses by marking them as
Confirmable (CON) messages. DOTS signal channel session
configuration requests and responses are marked as Confirmable (CON)
messages. As explained in Section 2.1 of [RFC7252], a Confirmable
message is retransmitted using a default timeout and exponential
back-off between retransmissions, until the DOTS server sends an
Acknowledgement message (ACK) with the same Message ID conveyed from
the DOTS client. Message transmission parameters are defined in

Section 4.8 of [RFC7252]. Reliability is provided to the responses
by marking them as Confirmable (CON) messages. The DOTS server can
either piggyback the response in the acknowledgement message or if
the DOTS server is not able to respond immediately to a request
carried in a Confirmable message, it simply responds with an Empty
Acknowledgement message so that the DOTS client can stop
retransmitting the request. Empty Acknowledgement message is
explained in Section 2.2 of [RFC7252]. When the response is ready,
the server sends it in a new Confirmable message which then in turn
needs to be acknowledged by the DOTS client (see Sections 5.2.1 and
Sections 5.2.2 of [RFC7252]). Requests and responses exchanged
between DOTS agents during peacetime are marked as Confirmable
messages.

Implementation Note: A DOTS client that receives a response in a CON
message may want to clean up the message state right after sending
the ACK. If that ACK is lost and the DOTS server retransmits the
CON, the DOTS client may no longer have any state to which to
correlate this response, making the retransmission an unexpected
message; the DOTS client will send a Reset message so it does not
receive any more retransmissions. This behavior is normal and not an
indication of an error (see Section 5.3.2 of [RFC7252] for more
details).

5.5.1. Discover Configuration Parameters

A GET request is used to obtain acceptable and current configuration
parameters on the DOTS server for DOTS signal channel session
configuration. Figure 14 shows how to obtain acceptable
configuration parameters for the server.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "version"
Uri-Path: "config"
```

Figure 14: GET to retrieve configuration

The DOTS server in the 2.05 (Content) response conveys the current,
minimum and maximum attribute values acceptable by the DOTS server.

```
   Content-Format: "application/cbor"
    {
      "heartbeat-interval": {
                             "CurrentValue": integer,
                             "MinValue": integer,
                             "MaxValue" : integer,
                            },
      "missing-hb-allowed": {
                             "CurrentValue": integer,
                             "MinValue": integer,
                             "MaxValue" : integer,
                            },
      "max-retransmit":     {
                             "CurrentValue": integer,
                             "MinValue": integer,
                             "MaxValue" : integer,
                            },
      "ack-timeout":        {
                             "CurrentValue": integer,
                             "MinValue": integer,
                             "MaxValue" : integer,
                            },
      "ack-random-factor":  {
                             "CurrentValue": number,
                             "MinValue": number,
                             "MaxValue" : number,
                            },
      "trigger-mitigation": {
                             "CurrentValue": boolean,
                            }
    }
```

                        Figure 15: GET response body

   Figure 16 shows an example of acceptable and current configuration
   parameters on the DOTS server for DOTS signal channel session
   configuration.

```
   Content-Format: "application/cbor"
   {
     "heartbeat-interval": {
                             "CurrentValue": 30,
                             "MinValue": 15,
                             "MaxValue" : 240,
                           },
       "missing-hb-allowed": {
                             "CurrentValue": 5,
                             "MinValue": 3,
                             "MaxValue" : 9,
                           },
       "max-retransmit":     {
                             "CurrentValue": 3,
                             "MinValue": 2,
                             "MaxValue" : 15,
                           },
       "ack-timeout":        {
                             "CurrentValue": 2,
                             "MinValue": 1,
                             "MaxValue" : 30,
                           },
       "ack-random-factor": {
                             "CurrentValue": 1.5,
                             "MinValue": 1.1,
                             "MaxValue" : 4.0,
                           },
       "trigger-mitigation": {
                             "CurrentValue": true,
                           }
   }
```

                  Figure 16: configuration response body

5.5.2. Convey DOTS Signal Channel Session Configuration

   A PUT request is used to convey the configuration parameters for the
   signaling channel (e.g., heartbeat interval, maximum
   retransmissions). Message transmission parameters for CoAP are
   defined in Section 4.8 of [RFC7252]. The RECOMMENDED values of
   transmission parameter values are ack_timeout (2 seconds), max-
   retransmit (3), ack-random-factor (1.5). In addition to those
   parameters, the RECOMMENDED specific DOTS transmission parameter
   values are heartbeat-interval (30 seconds) and missing-hb-allowed
   (5).

      Note: heartbeat-interval should be tweaked to also assist DOTS
      messages for NAT traversal (SIG-010 of

[I-D.ietf-dots-requirements]). According to [RFC8085], keepalive
messages must not be sent more frequently than once every 15
seconds and should use longer intervals when possible.
Furthermore, [RFC4787] recommends NATs to use a state timeout of 2
minutes or longer, but experience shows that sending packets every
15 to 30 seconds is necessary to prevent the majority of
middleboxes from losing state for UDP flows. From that
standpoint, this specification recommends a minimum heartbeat-
interval of 15 seconds and a maximum heartbeat-interval of 240
seconds. The recommended value of 30 seconds is selected to
anticipate the expiry of NAT state.

A heartbeat-interval of 30 second may be seen as too chatty in
some deployments. For such deployments, DOTS agents may negotiate
longer heartbeat-interval values to avoid overloading the network
with too frequent keepalives.

When a confirmable "CoAP ping" is sent, and if there is no response,
the "CoAP ping" will get retransmitted max-retransmit number of times
by the CoAP layer using an initial timeout set to a random duration
between ack_timeout and (ack-timeout*ack-random-factor) and
exponential back-off between retransmissions. By choosing the
recommended transmission parameters, the "CoAP ping" will timeout
after 45 seconds. If the DOTS agent does not receive any response
from the peer DOTS agent for missing-hb-allowed number of consecutive
"CoAP ping" confirmable messages, then it concludes that the DOTS
signal channel session is disconnected. A DOTS client MUST NOT
transmit a "CoAP ping" while waiting for the previous "CoAP ping"
response from the same DOTS server.

If the DOTS agent wishes to change the default values of message
transmission parameters, then it should follow the guidance given in
Section 4.8.1 of [RFC7252]. The DOTS agents MUST use the negotiated
values for message transmission parameters and default values for
non-negotiated message transmission parameters.

The signaling channel session configuration is applicable to a single
DOTS signal channel session between the DOTS agents.

```
   Header: PUT (Code=0.03)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "version"
   Uri-Path: "config"
   Content-Format: "application/cbor"
   {
    "signal-config": {
      "session-id": integer,
      "heartbeat-interval": integer,
      "missing-hb-allowed": integer,
      "max-retransmit": integer,
      "ack-timeout": integer,
      "ack-random-factor": number
      "trigger-mitigation": boolean
    }
   }
```

            Figure 17: PUT to convey the DOTS signal channel session
                            configuration data.

   The parameters are described below:

   session-id: Identifier for the DOTS signal channel session
      configuration data represented as an integer. This identifier
      MUST be generated by the DOTS client. This document does not make
      any assumption about how this identifier is generated. This is a
      mandatory attribute.

   heartbeat-interval: Time interval in seconds between two
      consecutive heartbeat messages. This is an optional attribute.

   missing-hb-allowed: Maximum number of consecutive heartbeat
      messages for which the DOTS agent did not receive a response
      before concluding that the session is disconnected. This is an
      optional attribute.

   max-retransmit: Maximum number of retransmissions for a message
      (referred to as MAX_RETRANSMIT parameter in CoAP). This is an
      optional attribute.

   ack-timeout: Timeout value in seconds used to calculate the initial
      retransmission timeout value (referred to as ACK_TIMEOUT parameter
      in CoAP). This is an optional attribute.

ack-random-factor: Random factor used to influence the timing of
   retransmissions (referred to as ACK_RANDOM_FACTOR parameter in
   CoAP). This is an optional attribute.

trigger-mitigation: If the parameter value is set to 'false', then
   DDoS mitigation is triggered only when the DOTS signal channel
   session is lost. Automated mitigation on loss of signal is
   discussed in Section 3.3.3 of [I-D.ietf-dots-architecture]. If
   the DOTS client ceases to respond to heartbeat messages, then the
   DOTS server can detect that the DOTS session is lost. The default
   value of the parameter is 'true'. This is an optional attribute.

In the PUT request at least one of the attributes heartbeat-interval,
missing-hb-allowed, max-retransmit, ack-timeout, ack-random-factor,
and trigger-mitigation MUST be present. The PUT request with higher
numeric session-id value over-rides the DOTS signal channel session
configuration data installed by a PUT request with a lower numeric
session-id value.

Figure 18 shows a PUT request example to convey the configuration
parameters for the DOTS signal channel.

```
Header: PUT (Code=0.03)
Uri-Host: "www.example.com"
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "v1"
Uri-Path: "config"
Content-Format: "application/cbor"
{
  "signal-config": {
   "session-id": 1234534333242,
   "heartbeat-interval": 91,
   "missing-hb-allowed": 3,
   "max-retransmit": 7,
   "ack-timeout": 5,
   "ack-random-factor": 1.5,
   "trigger-mitigation": false
  }
}
```

        Figure 18: PUT to convey the configuration parameters

The DOTS server indicates the result of processing the PUT request
using CoAP response codes:

o If the DOTS server finds the 'session-id' parameter value conveyed
   in the PUT request in its configuration data and if the DOTS

server has accepted the updated configuration parameters, then
2.04 (Changed) code is returned in the response.

o If the DOTS server does not find the 'session-id' parameter value
   conveyed in the PUT request in its configuration data and if the
   DOTS server has accepted the configuration parameters, then a
   response code 2.01 (Created) is returned in the response.

o If the request is missing one or more mandatory attributes, then
   4.00 (Bad Request) is returned in the response.

o If the request contains one or more invalid or unknown parameters,
   then 4.02 (Invalid query) code is returned in the response.

o Response code 4.22 (Unprocessable Entity) is returned in the
   response, if any of the heartbeat-interval, missing-hb-allowed,
   max-retransmit, target-protocol, ack-timeout, and ack-random-
   factor attribute values are not acceptable to the DOTS server.
   Upon receipt of the 4.22 error response code, the DOTS client
   should request the maximum and minimum attribute values acceptable
   to the DOTS server (Section 5.5.1). The DOTS client may re-try
   and send the PUT request with updated attribute values acceptable
   to the DOTS server.

5.5.3. Delete DOTS Signal Channel Session Configuration

   A DELETE request is used to delete the installed DOTS signal channel
   session configuration data (Figure 19).

      Header: DELETE (Code=0.04)
      Uri-Host: "host"
      Uri-Path: ".well-known"
      Uri-Path: "dots"
      Uri-Path: "version"
      Uri-Path: "config"
      Content-Format: "application/cbor"

                      Figure 19: DELETE configuration

   The DOTS server resets the DOTS signal channel session configuration
   back to the default values and acknowledges a DOTS client's request
   to remove the DOTS signal channel session configuration using 2.02
   (Deleted) response code.

5.6. Redirected Signaling

   Redirected Signaling is discussed in detail in Section 3.2.2 of
   [I-D.ietf-dots-architecture]. If the DOTS server wants to redirect
   the DOTS client to an alternative DOTS server for a signaling session
   then the response code 3.00 (alternate server) will be returned in
   the response to the client. The DOTS server can return the error
   response code 3.00 in response to a PUT request from the DOTS client
   or convey the error response code 3.00 in a unidirectional
   notification response from the DOTS server.

   The DOTS server in the error response conveys the alternate DOTS
   server FQDN, and the alternate DOTS server IP addresses and time to
   live values in the CBOR body.

```
{
    "alt-server": "string",
    "alt-server-record": [
      {
        "addr": "string",
        "ttl" : integer,
      }
    ]
}
```

                    Figure 20: Error response body

   The parameters are described below:

   alt-server: FQDN of an alternate DOTS server.

   addr: IP address of an alternate DOTS server.

   ttl: Time to live (TTL) represented as an integer number of seconds.

   Figure 21 shows a 3.00 response example to convey the DOTS alternate
   server www.example-alt.com, its IP addresses 2001:db8:6401::1 and
   2001:db8:6401::2, and TTL values 3600 and 1800.

```
{

    "alt-server": "www.example-alt.com",
    "alt-server-record": [
      {
        "ttl" :  3600,
        "addr": "2001:db8:6401::1"
      },
      {
        "ttl" :  1800,
        "addr": "2001:db8:6401::2"
      }
    ]
}
```

                 Figure 21: Example of error response body

   When the DOTS client receives 3.00 response, it considers the current
   request as having failed, but SHOULD try the request with the
   alternate DOTS server. During a DDOS attack, the DNS server may be
   subjected to DDOS attack, alternate DOTS server IP addresses conveyed
   in the 3.00 response help the DOTS client to skip DNS lookup of the
   alternate DOTS server and can try to establish UDP or TCP session
   with the alternate DOTS server IP addresses. The DOTS client SHOULD
   implement DNS64 function to handle the scenario where IPv6-only DOTS
   client communicates with IPv4-only alternate DOTS server.

5.7. Heartbeat Mechanism

   To provide a metric of signal health and distinguish an 'idle' signal
   channel from a 'disconnected' or 'defunct' session, the DOTS agent
   sends a heartbeat over the signal channel to maintain its half of the
   channel. The DOTS agent similarly expects a heartbeat from its peer
   DOTS agent, and may consider a session terminated in the extended
   absence of a peer agent heartbeat.

   While the communication between the DOTS agents is quiescent, the
   DOTS client will probe the DOTS server to ensure it has maintained
   cryptographic state and vice versa. Such probes can also keep alive
   firewall and/or NAT bindings. This probing reduces the frequency of
   establishing a new handshake when a DOTS signal needs to be conveyed
   to the DOTS server.

   In case of a volumetric DDoS attack saturating the incoming link to
   the DOTS client, all traffic from the DOTS server to the DOTS client
   will likely be dropped, although the DOTS server receives heartbeat
   requests and DOTS messages from the DOTS client. In this scenario,

the DOTS agents MUST behave differently to handle message
transmission and DOTS session liveliness during link saturation:

o The DOTS client MUST NOT consider the DOTS session terminated even
   after maximum "missing-hb-allowed" threshold is reached. The DOTS
   client SHOULD continue to use the current DOTS session, and send
   heartbeat requests over the current DOTS session, so the DOTS
   server knows the DOTS client has not disconnected the DOTS
   session. After the maximum "missing-hb-allowed" threshold is
   reached, the DOTS client SHOULD try (D)TLS session resumption.
   The DOTS client SHOULD send mitigation requests over the current
   DOTS session, and in parallel, try (D)TLS session resumption or
   0-RTT mode in DTLS 1.3 to piggyback the mitigation request in the
   ClientHello message. Once the link is no longer statured, if
   traffic from the DOTS server reaches the DOTS client over the
   current DOTS session, the DOTS client can stop (D)TLS session
   resumption or if (D)TLS session resumption is successful then
   disconnect the current DOTS session.

o If the DOTS server does not receive any traffic from the peer DOTS
   client, then the DOTS server sends heartbeat requests to the DOTS
   client and after maximum "missing-hb-allowed" threshold is
   reached, the DOTS server concludes the session is disconnected.

In DOTS over UDP, heartbeat messages may be exchanged between the
DOTS agents using the "COAP ping" mechanism defined in Section 4.2 of
[RFC7252]. Concretely, the DOTS agent sends an Empty Confirmable
message and the peer DOTS agent will respond by sending an Reset
message.

In DOTS over TCP, heartbeat messages can be exchanged between the
DOTS agents using the Ping and Pong messages specified in Section 4.4
of [I-D.ietf-core-coap-tcp-tls]. That is, the DOTS agent sends a
Ping message and the peer DOTS agent would respond by sending a
single Pong message.

6. Mapping parameters to CBOR

All parameters in the payload in the DOTS signal channel MUST be
mapped to CBOR types as follows and are given an integer key to save
space. The recipient of the payload MAY reject the information if it
is not suitably mapped.

```
/-------------------+----------------------+------------------------\
|  Parameter name   |  CBOR key            |  CBOR major type of value |
+-------------------+----------------------+------------------------+
|  mitigation-scope |  1                   |  5 (map)                |
|  scope            |  2                   |  5 (map)                |
|  mitigation-id    |  3                   |  0 (unsigned)           |
|  target-ip        |  4                   |  4 (array)              |
|  target-port-range|  5                   |  4                      |
|  lower-port       |  6                   |  0                      |
|  upper-port       |  7                   |  0                      |
|  target-protocol  |  8                   |  4                      |
|  fqdn             |  9                   |  4                      |
|  uri              |  10                  |  4                      |
|  alias-name       |  11                  |  4                      |
|  lifetime         |  12                  |  0                      |
|  attack-status    |  13                  |  0                      |
|  signal-config    |  14                  |  5                      |
|  heartbeat-interval| 15                  |  0                      |
|  max-retransmit   |  16                  |  0                      |
|  ack-timeout      |  17                  |  0                      |
|  ack-random-factor|  18                  |  7                      |
|  MinValue         |  19                  |  0                      |
|  MaxValue         |  20                  |  0                      |
|  status           |  21                  |  0                      |
|  bytes-dropped    |  22                  |  0                      |
|  bps-dropped      |  23                  |  0                      |
|  pkts-dropped     |  24                  |  0                      |
|  pps-dropped      |  25                  |  0                      |
|  session-id       |  26                  |  0                      |
|  trigger-mitigation| 27                  |  7 (simple types)       |
|  missing-hb-allowed|  28                 |  0                      |
|  CurrentValue     |  29                  |  0                      |
|  mitigation-start |  30                  |  7 (floating-point)     |
|  target-prefix    |  31                  |  4 (array)              |
|  client-identifier|  32                  |  2 (byte string)        |
|  alt-server       |  33                  |  2                      |
|  alt-server-record|  34                  |  4                      |
|  addr             |  35                  |  2                      |
|  ttl              |  36                  |  0                      |
\-------------------+----------------------+------------------------/
```

         Figure 22: CBOR mappings used in DOTS signal channel message

7. (D)TLS Protocol Profile and Performance considerations

   This section defines the (D)TLS protocol profile of DOTS signal
   channel over (D)TLS and DOTS data channel over TLS.

There are known attacks on (D)TLS, such as machine-in-the-middle and protocol downgrade. These are general attacks on (D)TLS and not specific to DOTS over (D)TLS; please refer to the (D)TLS RFCs for discussion of these security issues. DOTS agents MUST adhere to the (D)TLS implementation recommendations and security considerations of [RFC7525] except with respect to (D)TLS version. Since encryption of DOTS using (D)TLS is virtually a green-field deployment DOTS agents MUST implement only (D)TLS 1.2 or later.

Implementations compliant with this profile MUST implement all of the following items:

o DOTS agents MUST support DTLS record replay detection (Section 3.3 of [RFC6347]) to protect against replay attacks.

o DOTS client can use (D)TLS session resumption without server-side state [RFC5077] to resume session and convey the DOTS signal.

o Raw public keys [RFC7250] which reduce the size of the ServerHello, and can be used by servers that cannot obtain certificates (e.g., DOTS gateways on private networks).

Implementations compliant with this profile SHOULD implement all of the following items to reduce the delay required to deliver a DOTS signal:

o TLS False Start [RFC7918] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.

o Cached Information Extension [RFC7924] which avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake.

o TCP Fast Open [RFC7413] can reduce the number of round-trips to convey DOTS signal.

7.1. MTU and Fragmentation Issues

To avoid DOTS signal message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. If the Path MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP is used to convey the DOTS signal messages then the DOTS client must consider the amount of record expansion expected by the DTLS processing when calculating the size of CoAP message that fits within the path MTU. Path MTU MUST be greater than or equal to [CoAP

message size + DTLS overhead of 13 octets + authentication overhead
of the negotiated DTLS cipher suite + block padding (Section 4.1.1.1
of [RFC6347]]. If the request size exceeds the Path MTU then the
DOTS client MUST split the DOTS signal into separate messages, for
example the list of addresses in the 'target-ip' parameter could be
split into multiple lists and each list conveyed in a new PUT
request.

Implementation Note: DOTS choice of message size parameters works
well with IPv6 and with most of today's IPv4 paths. However, with
IPv4, it is harder to absolutely ensure that there is no IP
fragmentation. If IPv4 support on unusual networks is a
consideration and path MTU is unknown, implementations may want to
limit themselves to more conservative IPv4 datagram sizes such as 576
bytes, as per [RFC0791] IP packets up to 576 bytes should never need
to be fragmented, thus sending a maximum of 500 bytes of DOTS signal
over a UDP datagram will generally avoid IP fragmentation.

8. (D)TLS 1.3 considerations

TLS 1.3 [I-D.ietf-tls-tls13] provides critical latency improvements
for connection establishment over TLS 1.2. The DTLS 1.3 protocol
[I-D.rescorla-tls-dtls13] is based on the TLS 1.3 protocol and
provides equivalent security guarantees. (D)TLS 1.3 provides two
basic handshake modes of interest to DOTS signal channel:

o Absent packet loss, a full handshake in which the DOTS client is
   able to send the DOTS signal message after one round trip and the
   DOTS server immediately after receiving the first DOTS signal
   message from the client.

o 0-RTT mode in which the DOTS client can authenticate itself and
   send DOTS signal message on its first flight, thus reducing
   handshake latency. 0-RTT only works if the DOTS client has
   previously communicated with that DOTS server, which is very
   likely with the DOTS signal channel. The DOTS client SHOULD
   establish a (D)TLS session with the DOTS server during peacetime
   and share a PSK. During DDOS attack, the DOTS client can use the
   (D)TLS session to convey the DOTS signal message and if there is
   no response from the server after multiple re-tries then the DOTS
   client can resume the (D)TLS session in 0-RTT mode using PSK. A
   simplified TLS 1.3 handshake with 0-RTT DOTS signal message
   exchange is shown in Figure 23.

```
         DOTS Client                                    DOTS Server

        ClientHello
        (Finished)
        (0-RTT DOTS signal message)
        (end_of_early_data)          -------->
                                                          ServerHello
                                                 {EncryptedExtensions}
                                                 {ServerConfiguration}
                                                         {Certificate}
                                                   {CertificateVerify}
                                                            {Finished}
                                     <--------   [DOTS signal message]
        {Finished}                   -------->
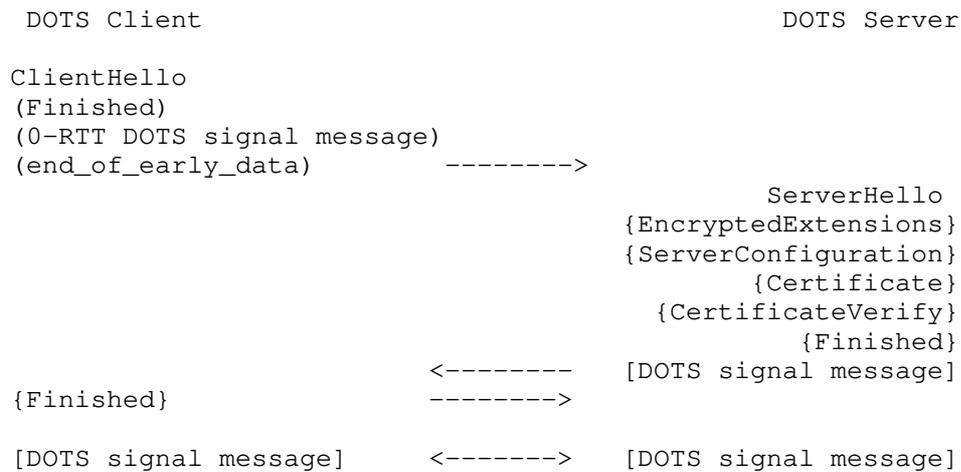
        [DOTS signal message]        <------->   [DOTS signal message]
```

                 Figure 23: TLS 1.3 handshake with 0-RTT

9. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

    (D)TLS based on client certificate can be used for mutual
    authentication between DOTS agents. If a DOTS gateway is involved,
    DOTS clients and DOTS gateway MUST perform mutual authentication;
    only authorized DOTS clients are allowed to send DOTS signals to a
    DOTS gateway. DOTS gateway and DOTS server MUST perform mutual
    authentication; DOTS server only allows DOTS signals from authorized
    DOTS gateway, creating a two-link chain of transitive authentication
    between the DOTS client and the DOTS server.

```
+----------------------------------------------+
|     example.com domain        +---------+    |
|                               |  AAA    |    |
|    +---------------+          | Server  |    |
|    | Application   |          +------+--+    |
|    | server        +                ^        |
|    | (DOTS client) |<---------------+  |   |              example.net
|    +---------------+          |    |   |   |              domain
|                               V    V   |
|                               +-----------+    |              +------------
--+                             |           | |  |              |
|    +-------------+            |           | |  |              |
|    |   Guest     +<-----x----->+          +<---------------->+   DOTS
|    | (DOTS client)|            |  DOTS     | |  |              |  Server
|    +-------------+             |  Gateway  | |  |              |
|                               +----+------+ |  |              +------------
--+                                  ^         |
|                                    |         |
|    +---------------+               |         |
|    | DDOS detector |               |         |
|    | (DOTS client) +<--------------+         |
|    +---------------+                         |
|                                              |
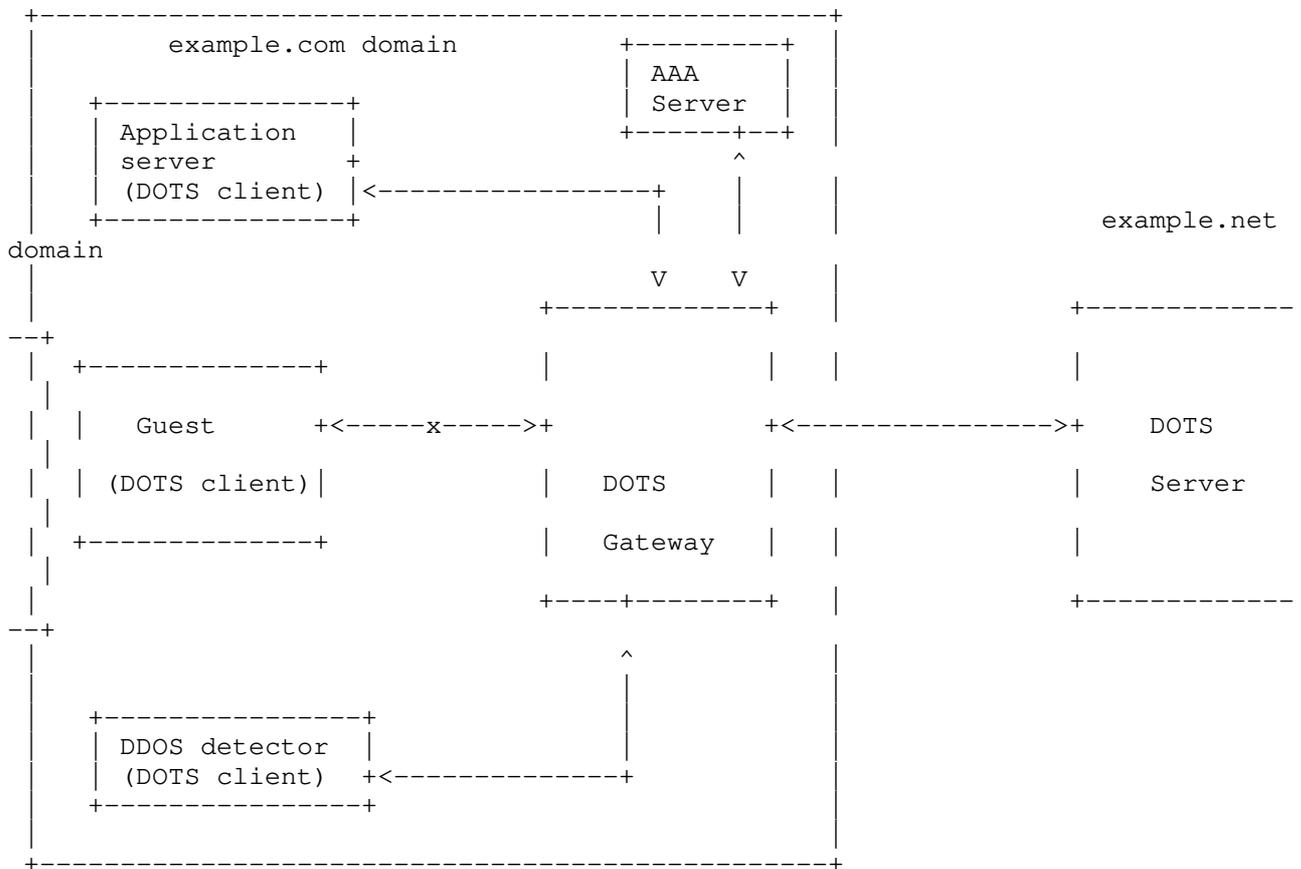+----------------------------------------------+
```

Figure 24: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 24, the DOTS gateway and DOTS
clients within the 'example.com' domain mutually authenticate with
each other. After the DOTS gateway validates the identity of a DOTS
client, it communicates with the AAA server in the 'example.com'
domain to determine if the DOTS client is authorized to request DDOS
mitigation. If the DOTS client is not authorized, a 4.01
(Unauthorized) is returned in the response to the DOTS client. In
this example, the DOTS gateway only allows the application server and
DDOS detector to request DDOS mitigation, but does not permit the
user of type 'guest' to request DDOS mitigation.

Also, DOTS gateway and DOTS server located in different domains MUST
perform mutual authentication (e.g., using certificates). A DOTS
server will only allow a DOTS gateway with a certificate for a
particular domain to request mitigation for that domain. In
reference to Figure 24, the DOTS server only allows the DOTS gateway
to request mitigation for 'example.com' domain and not for other
domains.

10. IANA Considerations

   This specification registers a default port, new URI suffix in the
   Well-Known URIs registry, new CoAP response code, new parameters for
   DOTS signal channel and establishes registries for mappings to CBOR.

10.1. DOTS Signal Channel UDP and TCP Port Number

   IANA has assigned the port number TBD to the DOTS signal channel
   protocol, for both UDP and TCP.

10.2. Well-Known 'dots' URI

   This memo registers the 'dots' well-known URI in the Well-Known URIs
   registry as defined by [RFC5785].

   URI suffix: dots

   Change controller: IETF

   Specification document(s): This RFC

   Related information: None

10.3. CoAP Response Code

   The following entry is added to the "CoAP Response Codes" sub-
   registry:

```
          +------+-----------------------------+-----------+
          | Code | Description                 | Reference |
          +------+-----------------------------+-----------+
          | 3.00 | Alternate server            | [RFCXXXX] |
          +------+-----------------------------+-----------+
```

                     Figure 25: CoAP Response Code

   [Note to RFC Editor: Please replace XXXX with the RFC number of this
   specification.]

10.4. DOTS signal channel CBOR Mappings Registry

   A new registry will be requested from IANA, entitled "DOTS signal
   channel CBOR Mappings Registry". The registry is to be created as
   Expert Review Required.

10.4.1. Registration Template

    Parameter name:
       Parameter names (e.g., "target_ip") in the DOTS signal channel.

    CBOR Key Value:
       Key value for the parameter. The key value MUST be an integer in
       the range of 1 to 65536. The key values in the range of 32768 to
       65536 are assigned for Vendor-Specific parameters.

    CBOR Major Type:
       CBOR Major type and optional tag for the claim.

    Change Controller:
       For Standards Track RFCs, list the "IESG". For others, give the
       name of the responsible party. Other details (e.g., postal
       address, email address, home page URI) may also be included.

    Specification Document(s):
       Reference to the document or documents that specify the parameter,
       preferably including URIs that can be used to retrieve copies of
       the documents. An indication of the relevant sections may also be
       included but is not required.

10.4.2. Initial Registry Contents

    o Parameter Name: "mitigation-scope"
    o CBOR Key Value: 1
    o CBOR Major Type: 5
    o Change Controller: IESG
    o Specification Document(s): this document

    o Parameter Name: "scope"
    o CBOR Key Value: 2
    o CBOR Major Type: 5
    o Change Controller: IESG
    o Specification Document(s): this document

    o Parameter Name: "mitigation-id"
    o CBOR Key Value: 3
    o CBOR Major Type: 0
    o Change Controller: IESG
    o Specification Document(s): this document

    o Parameter Name:target-ip
    o CBOR Key Value: 4
    o CBOR Major Type: 4
    o Change Controller: IESG

   o Specification Document(s): this document

   o Parameter Name: target-port-range
   o CBOR Key Value: 5
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: "lower-port"
   o CBOR Key Value: 6
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: "upper-port"
   o CBOR Key Value: 7
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: target-protocol
   o CBOR Key Value: 8
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: "fqdn"
   o CBOR Key Value: 9
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: "uri"
   o CBOR Key Value: 10
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: alias-name
   o CBOR Key Value: 11
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: "lifetime"
   o CBOR Key Value: 12
   o CBOR Major Type: 0
   o Change Controller: IESG

o Specification Document(s): this document

o Parameter Name: attack-status
o CBOR Key Value: 13
o CBOR Major Type: 0
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: signal-config
o CBOR Key Value: 14
o CBOR Major Type: 5
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: heartbeat-interval
o CBOR Key Value: 15
o CBOR Major Type: 0
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: max-retransmit
o CBOR Key Value: 16
o CBOR Major Type: 0
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: ack-timeout
o CBOR Key Value: 17
o CBOR Major Type: 0
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: ack-random-factor
o CBOR Key Value: 18
o CBOR Major Type: 7
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: MinValue
o CBOR Key Value: 19
o CBOR Major Type: 0
o Change Controller: IESG
o Specification Document(s): this document

o Parameter Name: MaxValue
o CBOR Key Value: 20
o CBOR Major Type: 0
o Change Controller: IESG

   o Specification Document(s): this document

   o Parameter Name: status
   o CBOR Key Value: 21
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: bytes-dropped
   o CBOR Key Value: 22
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: bps-dropped
   o CBOR Key Value: 23
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: pkts-dropped
   o CBOR Key Value: 24
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: pps-dropped
   o CBOR Key Value: 25
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: session-id
   o CBOR Key Value: 26
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: trigger-mitigation
   o CBOR Key Value: 27
   o CBOR Major Type: 7
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: missing-hb-allowed
   o CBOR Key Value: 28
   o CBOR Major Type: 0
   o Change Controller: IESG

   o Specification Document(s): this document

   o Parameter Name: CurrentValue
   o CBOR Key Value: 29
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:mitigation-start
   o CBOR Key Value: 30
   o CBOR Major Type: 7
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:target-prefix
   o CBOR Key Value: 31
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:client-identifier
   o CBOR Key Value: 32
   o CBOR Major Type: 2
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:alt-server
   o CBOR Key Value: 33
   o CBOR Major Type: 2
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:alt-server-record
   o CBOR Key Value: 34
   o CBOR Major Type: 4
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:addr
   o CBOR Key Value: 35
   o CBOR Major Type: 2
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name:ttl
   o CBOR Key Value: 36
   o CBOR Major Type: 0
   o Change Controller: IESG

      o Specification Document(s): this document

11. Implementation Status

      [Note to RFC Editor: Please remove this section and reference to
      [RFC7942] prior to publication.]

      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".

11.1. nttdots

   Organization: NTT Communication is developing a DOTS client and
      DOTS server software based on DOTS signal channel specified in
      this draft. It will be open-sourced.
   Description: Early implementation of DOTS protocol. It is aimed to
      implement a full DOTS protocol spec in accordance with maturing of
      DOTS protocol itself.
   Implementation: https://github.com/nttdots/go-dots
   Level of maturity: It is a early implementation of DOTS protocol.
      Messaging between DOTS clients and DOTS servers has been tested.
      Level of maturity will increase in accordance with maturing of
      DOTS protocol itself.
   Coverage: Capability of DOTS client: sending DOTS messages to the
      DOTS server in CoAP over DTLS as dots-signal. Capability of DOTS
      server: receiving dots-signal, validating received dots-signal,
      starting mitigation by handing over the dots-signal to DDOS
      mitigator.
   Licensing: It will be open-sourced with BSD 3-clause license.

   Implementation experience: It is implemented in Go-lang. Core
      specification of signaling is mature to be implemented, however,
      finding good libraries(like DTLS, CoAP) is rather difficult.
   Contact: Kaname Nishizuka <kaname@nttv6.jp>

12. Security Considerations

   Authenticated encryption MUST be used for data confidentiality and
   message integrity. (D)TLS based on client certificate MUST be used
   for mutual authentication. The interaction between the DOTS agents
   requires Datagram Transport Layer Security (DTLS) and Transport Layer
   Security (TLS) with a cipher suite offering confidentiality
   protection and the guidance given in [RFC7525] MUST be followed to
   avoid attacks on (D)TLS.

   A single DOTS signal channel between DOTS agents can be used to
   exchange multiple DOTS signal messages. To reduce DOTS client and
   DOTS server workload, DOTS client SHOULD re-use the (D)TLS session.

   If TCP is used between DOTS agents, an attacker may be able to inject
   RST packets, bogus application segments, etc., regardless of whether
   TLS authentication is used. Because the application data is TLS
   protected, this will not result in the application receiving bogus
   data, but it will constitute a DoS on the connection. This attack
   can be countered by using TCP-AO [RFC5925]. If TCP-AO is used, then
   any bogus packets injected by an attacker will be rejected by the
   TCP-AO integrity check and therefore will never reach the TLS layer.

   In order to prevent leaking internal information outside a client-
   domain, DOTS gateways located in the client-domain SHOULD NOT reveal
   the identity of internal DOTS clients (client-identifier) unless
   explicitly configured to do so.

   Special care should be taken in order to ensure that the activation
   of the proposed mechanism won't have an impact on the stability of
   the network (including connectivity and services delivered over that
   network).

   Involved functional elements in the cooperation system must establish
   exchange instructions and notification over a secure and
   authenticated channel. Adequate filters can be enforced to avoid
   that nodes outside a trusted domain can inject request such as
   deleting filtering rules. Nevertheless, attacks can be initiated
   from within the trusted domain if an entity has been corrupted.
   Adequate means to monitor trusted nodes should also be enabled.

13. Contributors

   The following individuals have contributed to this document:

   Mike Geller Cisco Systems, Inc. 3250 Florida 33309 USA Email:
   mgeller@cisco.com

   Robert Moskowitz HTT Consulting Oak Park, MI 42837 United States
   Email: rgm@htt-consult.com

   Dan Wing Email: dwing-ietf@fuggles.com

14. Acknowledgements

   Thanks to Christian Jacquenet, Roland Dobbins, Roman D. Danyliw,
   Michael Richardson, Ehud Doron, Kaname Nishizuka, Dave Dolson, Liang
   Xia, Jon Shallow, and Gilbert Clark for the discussion and comments.

15. References

15.1. Normative References

   [I-D.ietf-core-coap-tcp-tls]
              Bormann, C., Lemay, S., Tschofenig, H., Hartke, K.,
              Silverajan, B., and B. Raymor, "CoAP (Constrained
              Application Protocol) over TCP, TLS, and WebSockets",
              draft-ietf-core-coap-tcp-tls-10 (work in progress),
              October 2017.

   [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>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
              Uniform Resource Identifiers (URIs)", RFC 5785,
              DOI 10.17487/RFC5785, April 2010,
              <https://www.rfc-editor.org/info/rfc5785>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
              Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
              June 2010, <https://www.rfc-editor.org/info/rfc5925>.

   [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>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
              Weiler, S., and T. Kivinen, "Using Raw Public Keys in
              Transport Layer Security (TLS) and Datagram Transport
              Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
              June 2014, <https://www.rfc-editor.org/info/rfc7250>.

   [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252,
              DOI 10.17487/RFC7252, June 2014,
              <https://www.rfc-editor.org/info/rfc7252>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <https://www.rfc-editor.org/info/rfc7641>.

15.2. Informative References

   [I-D.ietf-core-comi]
              Veillette, M., Stok, P., Pelov, A., and A. Bierman, "CoAP
              Management Interface", draft-ietf-core-comi-01 (work in
              progress), July 2017.

   [I-D.ietf-core-yang-cbor]
              Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A.
              Minaburo, "CBOR Encoding of Data Modeled with YANG",
              draft-ietf-core-yang-cbor-05 (work in progress), August
              2017.

   [I-D.ietf-dots-architecture]
              Mortensen, A., Andreasen, F., Reddy, T.,
              christopher_gray3@cable.comcast.com, c., Compton, R., and
              N. Teague, "Distributed-Denial-of-Service Open Threat
              Signaling (DOTS) Architecture", draft-ietf-dots-
              architecture-05 (work in progress), October 2017.

   [I-D.ietf-dots-data-channel]
              Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., Patil,
              P., Mortensen, A., and N. Teague, "Distributed Denial-of-
              Service Open Threat Signaling (DOTS) Data Channel", draft-
              ietf-dots-data-channel-06 (work in progress), October
              2017.

   [I-D.ietf-dots-requirements]
              Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed
              Denial of Service (DDoS) Open Threat Signaling
              Requirements", draft-ietf-dots-requirements-07 (work in
              progress), October 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-09 (work
              in progress), November 2017.

   [I-D.ietf-tls-tls13]
              Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", draft-ietf-tls-tls13-21 (work in progress),
              July 2017.

   [I-D.rescorla-tls-dtls13]
              Rescorla, E., Tschofenig, H., and N. Modadugu, "The
              Datagram Transport Layer Security (DTLS) Protocol Version
              1.3", draft-rescorla-tls-dtls13-01 (work in progress),
              March 2017.

   [proto_numbers]
              "IANA, "Protocol Numbers"", 2011,
              <http://www.iana.org/assignments/protocol-numbers>.

   [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
              DOI 10.17487/RFC0791, September 1981,
              <https://www.rfc-editor.org/info/rfc791>.

   [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
              Congestion Control Protocol (DCCP)", RFC 4340,
              DOI 10.17487/RFC4340, March 2006,
              <https://www.rfc-editor.org/info/rfc4340>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
              (CIDR): The Internet Address Assignment and Aggregation
              Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
              2006, <https://www.rfc-editor.org/info/rfc4632>.

   [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
              Denial-of-Service Considerations", RFC 4732,
              DOI 10.17487/RFC4732, December 2006,
              <https://www.rfc-editor.org/info/rfc4732>.

   [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address
              Translation (NAT) Behavioral Requirements for Unicast
              UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
              2007, <https://www.rfc-editor.org/info/rfc4787>.

   [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol",
              RFC 4960, DOI 10.17487/RFC4960, September 2007,
              <https://www.rfc-editor.org/info/rfc4960>.

   [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common
              Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007,
              <https://www.rfc-editor.org/info/rfc4987>.

   [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig,
              "Transport Layer Security (TLS) Session Resumption without
              Server-Side State", RFC 5077, DOI 10.17487/RFC5077,
              January 2008, <https://www.rfc-editor.org/info/rfc5077>.

   [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <https://www.rfc-editor.org/info/rfc6020>.

   [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>.

   [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
              "Default Address Selection for Internet Protocol Version 6
              (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
              <https://www.rfc-editor.org/info/rfc6724>.

   [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
              "Enrollment over Secure Transport", RFC 7030,
              DOI 10.17487/RFC7030, October 2013,
              <https://www.rfc-editor.org/info/rfc7030>.

   [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object
              Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
              October 2013, <https://www.rfc-editor.org/info/rfc7049>.

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

   [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the
              NETCONF Protocol over Transport Layer Security (TLS) with
              Mutual X.509 Authentication", RFC 7589,
              DOI 10.17487/RFC7589, June 2015,
              <https://www.rfc-editor.org/info/rfc7589>.

   [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport
              Layer Security (TLS) False Start", RFC 7918,
              DOI 10.17487/RFC7918, August 2016,
              <https://www.rfc-editor.org/info/rfc7918>.

   [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security
              (TLS) Cached Information Extension", RFC 7924,
              DOI 10.17487/RFC7924, July 2016,
              <https://www.rfc-editor.org/info/rfc7924>.

   [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>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

Authors' Addresses

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

   Email: kondtir@gmail.com

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

   Prashanth Patil
   Cisco Systems, Inc.

   Email: praspati@cisco.com

   Andrew Mortensen
   Arbor Networks, Inc.
   2727 S. State St
   Ann Arbor, MI 48104
   United States

   Email: amortensen@arbor.net

   Nik Teague
   Verisign, Inc.
   United States

   Email: nteague@verisign.com

        Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
                            Channel Specification
                      draft-ietf-dots-signal-channel-37

   Abstract

      This document specifies the DOTS signal channel, a protocol for
      signaling the need for protection against Distributed Denial-of-
      Service (DDoS) attacks to a server capable of enabling network
      traffic mitigation on behalf of the requesting client.

      A companion document defines the DOTS data channel, a separate
      reliable communication layer for DOTS management and configuration
      purposes.

   Editorial Note (To be removed by RFC Editor)

      Please update these statements within the document with the RFC
      number to be assigned to this document:

      o "This version of this YANG module is part of RFC XXXX;"

      o "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling
         (DOTS) Signal Channel Specification";

      o "| [RFCXXXX] |"

      o reference: RFC XXXX

      Please update this statement with the RFC number to be assigned to
      the following documents:

      o "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling
         (DOTS) Data Channel Specification (used to be I-D.ietf-dots-data-
         channel)

Please update TBD/TBD1/TBD2 statements with the assignments made by
IANA to DOTS Signal Channel Protocol.

Also, please update the "revision" date of the YANG modules.

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 29, 2020.

Copyright Notice

Table of Contents

1. Introduction

   A distributed denial-of-service (DDoS) attack is a distributed
   attempt to make machines or network resources unavailable to their
   intended users. In most cases, sufficient scale for an effective
   attack can be achieved by compromising enough end-hosts and using
   those infected hosts to perpetrate and amplify the attack. The
   victim in this attack can be an application server, a host, a router,
   a firewall, or an entire network.

   Network applications have finite resources like CPU cycles, the
   number of processes or threads they can create and use, the maximum
   number of simultaneous connections they can handle, the limited
   resources of the control plane, etc. When processing network
   traffic, such applications are supposed to use these resources to
   provide the intended functionality in the most efficient manner.
   However, a DDoS attacker may be able to prevent an application from
   performing its intended task by making the application exhaust its
   finite resources.

   A TCP DDoS SYN-flood [RFC4987], for example, is a memory-exhausting
   attack while an ACK-flood is a CPU-exhausting attack. Attacks on the
   link are carried out by sending enough traffic so that the link
   becomes congested, thereby likely causing packet loss for legitimate
   traffic. Stateful firewalls can also be attacked by sending traffic
   that causes the firewall to maintain an excessive number of states
   that may jeopardize the firewall's operation overall, besides likely
   performance impacts. The firewall then runs out of memory, and can
   no longer instantiate the states required to process legitimate
   flows. Other possible DDoS attacks are discussed in [RFC4732].

   In many cases, it may not be possible for network administrators to
   determine the cause(s) of an attack. They may instead just realize
   that certain resources seem to be under attack. This document
   defines a lightweight protocol that allows a DOTS client to request
   mitigation from one or more DOTS servers for protection against
   detected, suspected, or anticipated attacks. This protocol enables
   cooperation between DOTS agents to permit a highly-automated network
   defense that is robust, reliable, and secure. Note that "secure"
   means the support of the features defined in Section 2.4 of
   [RFC8612].

   An example of a network diagram that illustrates a deployment of DOTS
   agents is shown in Figure 1. In this example, a DOTS server is
   operating on the access network. A DOTS client is located on the LAN
   (Local Area Network), while a DOTS gateway is embedded in the CPE
   (Customer Premises Equipment).

```
      Network
      Resource              CPE router          Access network     _____
    +----------+        +--------------+      +-------------+     /          \
    |          |        |              |      |             |    |            |
    |DOTS client|_____|  DOTS gateway |_____|  DOTS server |___| Internet  |
    |          |        |              |      |             |    |            |
    +----------+        +--------------+      +-------------+     _____/
```

                 Figure 1: Sample DOTS Deployment (1)

   DOTS servers can also be reachable over the Internet, as depicted in
   Figure 2.

```
      Network                                                   DDoS mitigation
      Resource              CPE router          _____          service
    +----------+        +--------------+       /          \     +-------------+
    |          |        |              |      |            |     |             |
    |DOTS client|_____|DOTS gateway   |_____| Internet   |____| DOTS server |
    |          |        |              |      |            |     |             |
    +----------+        +--------------+       _____/      +-------------+
```

                 Figure 2: Sample DOTS Deployment (2)

   In typical deployments, the DOTS client belongs to a different
   administrative domain than the DOTS server. For example, the DOTS
   client is embedded in a firewall protecting services owned and
   operated by a customer, while the DOTS server is owned and operated
   by a different administrative entity (service provider, typically)
   providing DDoS mitigation services. The latter might or might not
   provide connectivity services to the network hosting the DOTS client.

   The DOTS server may (not) be co-located with the DOTS mitigator. In
   typical deployments, the DOTS server belongs to the same
   administrative domain as the mitigator. The DOTS client can
   communicate directly with a DOTS server or indirectly via a DOTS
   gateway.

   The document adheres to the DOTS architecture
   [I-D.ietf-dots-architecture]. The requirements for DOTS signal
   channel protocol are documented in [RFC8612]. This document
   satisfies all the use cases discussed in [I-D.ietf-dots-use-cases].

   This document focuses on the DOTS signal channel. This is a
   companion document of the DOTS data channel specification
   [I-D.ietf-dots-data-channel] that defines a configuration and a bulk
   data exchange mechanism supporting the DOTS signal channel.

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.

(D)TLS is used for statements that apply to both Transport Layer
Security [RFC5246][RFC8446] and Datagram Transport Layer Security
[RFC6347]. Specific terms are used for any statement that applies to
either protocol alone.

The reader should be familiar with the terms defined in [RFC8612].

The meaning of the symbols in YANG tree diagrams is defined in
[RFC8340].

3. Design Overview

The DOTS signal channel is built on top of the Constrained
Application Protocol (CoAP) [RFC7252], a lightweight protocol
originally designed for constrained devices and networks. The many
features of CoAP (expectation of packet loss, support for
asynchronous Non-confirmable messaging, congestion control, small
message overhead limiting the need for fragmentation, use of minimal
resources, and support for (D)TLS) makes it a good candidate to build
the DOTS signaling mechanism from.

The DOTS signal channel is layered on existing standards (Figure 3).

```
+--------------------+
| DOTS Signal Channel |
+--------------------+
|        CoAP        |
+---------+----------+
|   TLS   |   DTLS   |
+---------+----------+
|   TCP   |   UDP    |
+---------+----------+
|         IP         |
+--------------------+
```

Figure 3: Abstract Layering of DOTS Signal Channel over CoAP over
(D)TLS

In some cases, a DOTS client and server may have mutual agreement to
use a specific port number, such as by explicit configuration or

dynamic discovery [I-D.ietf-dots-server-discovery]. Absent such
mutual agreement, the DOTS signal channel MUST run over port number
TBD as defined in Section 9.1, for both UDP and TCP. In order to use
a distinct port number (as opposed to TBD), DOTS clients and servers
SHOULD support a configurable parameter to supply the port number to
use.

> Note: The rationale for not using the default port number 5684
> ((D)TLS CoAP) is to avoid the discovery of services and resources
> discussed in [RFC7252] and allow for differentiated behaviors in
> environments where both a DOTS gateway and an IoT gateway (e.g.,
> Figure 3 of [RFC7452]) are co-located.
>
> Particularly, the use of a default port number is meant to
> simplify DOTS deployment in scenarios where no explicit IP address
> configuration is required. For example, the use of the default
> router as DOTS server aims to ease DOTS deployment within LANs (in
> which, CPEs embed a DOTS gateway as illustrated in Figures 1 and
> 2) without requiring a sophisticated discovery method and
> configuration tasks within the LAN. The use of anycast is meant
> to simplify DOTS client configuration, including service
> discovery. In such anycast-based scenario, a DOTS client
> initiating a DOTS session to the DOTS server anycast address may,
> for example, be (1) redirected to the DOTS server unicast address
> to be used by the DOTS client following the procedure discussed in
> Section 4.6 or (2) relayed to a unicast DOTS server.

The signal channel uses the "coaps" URI scheme defined in Section 6
of [RFC7252] and the "coaps+tcp" URI scheme defined in Section 8.2 of
[RFC8323] to identify DOTS server resources accessible using CoAP
over UDP secured with DTLS and CoAP over TCP secured with TLS,
respectively.

The DOTS signal channel can be established between two DOTS agents
prior or during an attack. The DOTS signal channel is initiated by
the DOTS client. The DOTS client can then negotiate, configure, and
retrieve the DOTS signal channel session behavior with its DOTS peer
(Section 4.5). Once the signal channel is established, the DOTS
agents may periodically send heartbeats to keep the channel active
(Section 4.7). At any time, the DOTS client may send a mitigation
request message (Section 4.4) to a DOTS server over the active signal
channel. While mitigation is active (because of the higher
likelihood of packet loss during a DDoS attack), the DOTS server
periodically sends status messages to the client, including basic
mitigation feedback details. Mitigation remains active until the
DOTS client explicitly terminates mitigation, or the mitigation
lifetime expires. Also, the DOTS server may rely on the signal

channel session loss to trigger mitigation for pre-configured
mitigation requests (if any).

DOTS signaling can happen with DTLS over UDP and TLS over TCP.
Likewise, DOTS requests may be sent using IPv4 or IPv6 transfer
capabilities. A Happy Eyeballs procedure for DOTS signal channel is
specified in Section 4.3.

A DOTS client is entitled to access only to resources it creates. In
particular, a DOTS client can not retrieve data related to mitigation
requests created by other DOTS clients of the same DOTS client
domain.

Messages exchanged between DOTS agents are serialized using Concise
Binary Object Representation (CBOR) [RFC7049], a binary encoding
scheme designed for small code and message size. CBOR-encoded
payloads are used to carry signal channel-specific payload messages
which convey request parameters and response information such as
errors. In order to allow reusing data models across protocols,
[RFC7951] specifies the JavaScript Object Notation (JSON) encoding of
YANG-modeled data. A similar effort for CBOR is defined in
[I-D.ietf-core-yang-cbor].

DOTS agents determine that a CBOR data structure is a DOTS signal
channel object from the application context, such as from the port
number assigned to the DOTS signal channel. The other method DOTS
agents use to indicate that a CBOR data structure is a DOTS signal
channel object is the use of the "application/dots+cbor" content type
(Section 9.3).

This document specifies a YANG module for representing DOTS
mitigation scopes, DOTS signal channel session configuration data,
and DOTS redirected signaling (Section 5). All parameters in the
payload of the DOTS signal channel are mapped to CBOR types as
specified in Table 4 (Section 6).

In order to prevent fragmentation, DOTS agents must follow the
recommendations documented in Section 4.6 of [RFC7252]. Refer to
Section 7.3 for more details.

DOTS agents MUST support GET, PUT, and DELETE CoAP methods. The
payload included in CoAP responses with 2.xx Response Codes MUST be
of content type "application/dots+cbor". CoAP responses with 4.xx
and 5.xx error Response Codes MUST include a diagnostic payload
(Section 5.5.2 of [RFC7252]). The Diagnostic Payload may contain
additional information to aid troubleshooting.

In deployments where multiple DOTS clients are enabled in a network
(owned and operated by the same entity), the DOTS server may detect
conflicting mitigation requests from these clients. This document
does not aim to specify a comprehensive list of conditions under
which a DOTS server will characterize two mitigation requests from
distinct DOTS clients as conflicting, nor recommend a DOTS server
behavior for processing conflicting mitigation requests. Those
considerations are implementation- and deployment-specific.
Nevertheless, the document specifies the mechanisms to notify DOTS
clients when conflicts occur, including the conflict cause
(Section 4.4).

In deployments where one or more translators (e.g., Traditional NAT
[RFC3022], CGN [RFC6888], NAT64 [RFC6146], NPTv6 [RFC6296]) are
enabled between the client's network and the DOTS server, DOTS signal
channel messages forwarded to a DOTS server MUST NOT include internal
IP addresses/prefixes and/or port numbers; external addresses/
prefixes and/or port numbers as assigned by the translator MUST be
used instead. This document does not make any recommendation about
possible translator discovery mechanisms. The following are some
(non-exhaustive) deployment examples that may be considered:

o Port Control Protocol (PCP) [RFC6887] or Session Traversal
   Utilities for NAT (STUN) [RFC5389] may be used to retrieve the
   external addresses/prefixes and/or port numbers. Information
   retrieved by means of PCP or STUN will be used to feed the DOTS
   signal channel messages that will be sent to a DOTS server.

o A DOTS gateway may be co-located with the translator. The DOTS
   gateway will need to update the DOTS messages, based upon the
   local translator's binding table.

4. DOTS Signal Channel: Messages & Behaviors

4.1. DOTS Server(s) Discovery

   This document assumes that DOTS clients are provisioned with the
   reachability information of their DOTS server(s) using any of a
   variety of means (e.g., local configuration, or dynamic means such as
   DHCP [I-D.ietf-dots-server-discovery]). The description of such
   means is out of scope of this document.

   Likewise, it is out of scope of this document to specify the behavior
   to be followed by a DOTS client to send DOTS requests when multiple
   DOTS servers are provisioned (e.g., contact all DOTS servers, select
   one DOTS server among the list). Such behavior is specified in other
   documents (e.g., [I-D.ietf-dots-multihoming]).

4.2. CoAP URIs

   The DOTS server MUST support the use of the path-prefix of "/.well-
   known/" as defined in [RFC5785] and the registered name of "dots".
   Each DOTS operation is indicated by a path-suffix that indicates the
   intended operation. The operation path (Table 1) is appended to the
   path-prefix to form the URI used with a CoAP request to perform the
   desired DOTS operation.

```
      +-----------------------+----------------+-------------+
      | Operation             | Operation Path | Details     |
      +-----------------------+----------------+-------------+
      | Mitigation            | /mitigate      | Section 4.4 |
      +-----------------------+----------------+-------------+
      | Session configuration | /config        | Section 4.5 |
      +-----------------------+----------------+-------------+
```

               Table 1: Operations and their Corresponding URIs

4.3. Happy Eyeballs for DOTS Signal Channel

   [RFC8612] mentions that DOTS agents will have to support both
   connectionless and connection-oriented protocols. As such, the DOTS
   signal channel is designed to operate with DTLS over UDP and TLS over
   TCP. Further, a DOTS client may acquire a list of IPv4 and IPv6
   addresses (Section 4.1), each of which can be used to contact the
   DOTS server using UDP and TCP. If no list of IPv4 and IPv6 addresses
   to contact the DOTS server is configured (or discovered), the DOTS
   client adds the IPv4/IPv6 addresses of its default router to the
   candidate list to contact the DOTS server.

   The following specifies the procedure to follow to select the address
   family and the transport protocol for sending DOTS signal channel
   messages.

   Such procedure is needed to avoid experiencing long connection
   delays. For example, if an IPv4 path to reach a DOTS server is
   functional, but the DOTS server's IPv6 path is non-functional, a
   dual-stack DOTS client may experience a significant connection delay
   compared to an IPv4-only DOTS client, in the same network conditions.
   The other problem is that if a middlebox between the DOTS client and
   DOTS server is configured to block UDP traffic, the DOTS client will
   fail to establish a DTLS association with the DOTS server and, as a
   consequence, will have to fall back to TLS over TCP, thereby
   incurring significant connection delays.

   To overcome these connection setup problems, the DOTS client attempts
   to connect to its DOTS server(s) using both IPv6 and IPv4, and tries

both DTLS over UDP and TLS over TCP following a DOTS Happy Eyeballs
approach. To some extent, this approach is similar to the Happy
Eyeballs mechanism defined in [RFC8305]. The connection attempts are
performed by the DOTS client when it initializes, or in general when
it has to select an address family and transport to contact its DOTS
server. The results of the Happy Eyeballs procedure are used by the
DOTS client for sending its subsequent messages to the DOTS server.
The difference in behavior with respect to the Happy Eyeballs
mechanism [RFC8305] are listed below:

o   The order of preference of the DOTS signal channel address family
    and transport protocol (most preferred first) is: UDP over IPv6,
    UDP over IPv4, TCP over IPv6, and finally TCP over IPv4. This
    order adheres to the address preference order specified in
    [RFC6724] and the DOTS signal channel preference which privileges
    the use of UDP over TCP (to avoid TCP's head of line blocking).

o   The DOTS client after successfully establishing a connection MUST
    cache information regarding the outcome of each connection attempt
    for a specific time period, and it uses that information to avoid
    thrashing the network with subsequent attempts. The cached
    information is flushed when its age exceeds a specific time period
    on the order of few minutes (e.g., 10 min). Typically, if the
    DOTS client has to re-establish the connection with the same DOTS
    server within few seconds after the Happy Eyeballs mechanism is
    completed, caching avoids trashing the network especially in the
    presence of DDoS attack traffic.

o   If DOTS signal channel session is established with TLS (but DTLS
    failed), the DOTS client periodically repeats the mechanism to
    discover whether DOTS signal channel messages with DTLS over UDP
    becomes available from the DOTS server, so the DOTS client can
    migrate the DOTS signal channel from TCP to UDP. Such probing
    SHOULD NOT be done more frequently than every 24 hours and MUST
    NOT be done more frequently than every 5 minutes.

When connection attempts are made during an attack, the DOTS client
SHOULD use a "Connection Attempt Delay" [RFC8305] set to 100 ms.

In reference to Figure 4, the DOTS client proceeds with the
connection attempts following the rules in [RFC8305]. In this
example, it is assumed that the IPv6 path is broken and UDP traffic
is dropped by a middlebox but has little impact to the DOTS client
because there is no long delay before using IPv4 and TCP.

```
        +-----------+                              +-----------+
        |DOTS client|                              |DOTS server|
        +-----------+                              +-----------+
              |                                          |
      T0  |--DTLS ClientHello, IPv6 ---->X                |
      T1  |--DTLS ClientHello, IPv4 ---->X                |
      T2  |--TCP SYN, IPv6-------------->X                |
      T3  |--TCP SYN, IPv4------------------------------------------->|
          |<-TCP SYNACK----------------------------------------------|
          |--TCP ACK------------------------------------------------->|
          |<-----------Establish TLS Session----------------------->|
          |---------------DOTS signal------------------------------->|
              |                                          |
```

```
   Note:
    * Retransmission messages are not shown.
    * T1-T0=T2-T1=T3-T2= Connection Attempt Delay.
```

                    Figure 4: DOTS Happy Eyeballs (Sample Flow)

   A single DOTS signal channel between DOTS agents can be used to
   exchange multiple DOTS signal messages. To reduce DOTS client and
   DOTS server workload, DOTS clients SHOULD re-use the (D)TLS session.

4.4. DOTS Mitigation Methods

   The following methods are used by a DOTS client to request, withdraw,
   or retrieve the status of mitigation requests:

   PUT: DOTS clients use the PUT method to request mitigation from a
            DOTS server (Section 4.4.1). During active mitigation, DOTS
            clients may use PUT requests to carry mitigation efficacy
            updates to the DOTS server (Section 4.4.3).

   GET: DOTS clients may use the GET method to subscribe to DOTS
            server status messages, or to retrieve the list of its
            mitigations maintained by a DOTS server (Section 4.4.2).

   DELETE: DOTS clients use the DELETE method to withdraw a request for
            mitigation from a DOTS server (Section 4.4.4).

   Mitigation request and response messages are marked as Non-
   confirmable messages (Section 2.2 of [RFC7252]).

   DOTS agents MUST NOT send more than one UDP datagram per round-trip
   time (RTT) to the peer DOTS agent on average following the data
   transmission guidelines discussed in Section 3.1.3 of [RFC8085].

Requests marked by the DOTS client as Non-confirmable messages are
sent at regular intervals until a response is received from the DOTS
server. If the DOTS client cannot maintain an RTT estimate, it MUST
NOT send more than one Non-confirmable request every 3 seconds, and
SHOULD use an even less aggressive rate whenever possible (case 2 in
Section 3.1.3 of [RFC8085]).

JSON encoding of YANG modelled data [RFC7951] is used to illustrate
the various methods defined in the following sub-sections. Also, the
examples use the Labels defined in Sections 9.6.2, 9.6.3, 9.6.4, and
9.6.5.

4.4.1. Request Mitigation

When a DOTS client requires mitigation for some reason, the DOTS
client uses the CoAP PUT method to send a mitigation request to its
DOTS server(s) (Figures 5 and 6).

If a DOTS client is entitled to solicit the DOTS service, the DOTS
server enables mitigation on behalf of the DOTS client by
communicating the DOTS client's request to a mitigator (which may be
co-located with the DOTS server) and relaying the feedback of the
thus-selected mitigator to the requesting DOTS client.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=123"
Content-Format: "application/dots+cbor"

{
  ...
}
```

               Figure 5: PUT to Convey DOTS Mitigation Requests

The order of the Uri-Path options is important as it defines the CoAP
resource. In particular, 'mid' MUST follow 'cuid'.

The additional Uri-Path parameters to those defined in Section 4.2
are as follows:

cuid: Stands for Client Unique Identifier. A globally unique
      identifier that is meant to prevent collisions among DOTS
      clients, especially those from the same domain. It MUST be
      generated by DOTS clients.

For the reasons discussed in Appendix A, implementations SHOULD
set 'cuid' using the following procedure: first, the
Distinguished Encoding Rules (DER)-encoded Abstract Syntax
Notation One (ASN.1) representation of the Subject Public Key
Info (SPKI) of the DOTS client X.509 certificate [RFC5280], the
DOTS client raw public key [RFC7250], the "Pre-Shared Key (PSK)
identity" used by the DOTS client in the TLS 1.2
ClientKeyExchange message, or the "identity" used by the DOTS
client in the "pre_shared_key" TLS 1.3 extension is input to
the SHA-256 [RFC6234] cryptographic hash. Then, the output of
the cryptographic hash algorithm is truncated to 16 bytes;
truncation is done by stripping off the final 16 bytes. The
truncated output is base64url encoded (Section 5 of [RFC4648])
with the trailing "=" removed from the encoding, and the
resulting value used as the 'cuid'.

The 'cuid' is intended to be stable when communicating with a
given DOTS server, i.e., the 'cuid' used by a DOTS client
SHOULD NOT change over time. Distinct 'cuid' values MAY be
used by a single DOTS client per DOTS server.

If a DOTS client has to change its 'cuid' for some reason, it
MUST NOT do so when mitigations are still active for the old
'cuid'. The 'cuid' SHOULD be 22 characters to avoid DOTS
signal message fragmentation over UDP. Furthermore, if that
DOTS client created aliases and filtering entries at the DOTS
server by means of the DOTS data channel, it MUST delete all
the entries bound to the old 'cuid' and re-install them using
the new 'cuid'.

DOTS servers MUST return 4.09 (Conflict) error code to a DOTS
peer to notify that the 'cuid' is already in-use by another
DOTS client. Upon receipt of that error code, a new 'cuid'
MUST be generated by the DOTS peer (e.g., using [RFC4122]).

Client-domain DOTS gateways MUST handle 'cuid' collision
directly and it is RECOMMENDED that 'cuid' collision is handled
directly by server-domain DOTS gateways.

DOTS gateways MAY rewrite the 'cuid' used by peer DOTS clients.
Triggers for such rewriting are out of scope.

This is a mandatory Uri-Path parameter.

mid: Identifier for the mitigation request represented with an
      integer. This identifier MUST be unique for each mitigation
      request bound to the DOTS client, i.e., the 'mid' parameter
      value in the mitigation request needs to be unique (per 'cuid'

and DOTS server) relative to the 'mid' parameter values of
active mitigation requests conveyed from the DOTS client to the
DOTS server.

In order to handle out-of-order delivery of mitigation
requests, 'mid' values MUST increase monotonically.

If the 'mid' value has reached 3/4 of (2**32 - 1) (i.e.,
3221225471) and no attack is detected, the DOTS client MUST
reset 'mid' to 0 to handle 'mid' rollover. If the DOTS client
maintains mitigation requests with pre-configured scopes, it
MUST re-create them with the 'mid' restarting at 0.

This identifier MUST be generated by the DOTS client.

This is a mandatory Uri-Path parameter.

'cuid' and 'mid' MUST NOT appear in the PUT request message body
(Figure 6). The schema in Figure 6 uses the types defined in
Section 6. Note that this figure (and other similar figures
depicting a schema) are non-normative sketches of the structure of
the message.

```
  {
    "ietf-dots-signal-channel:mitigation-scope": {
      "scope": [
        {
          "target-prefix": [
            "string"
           ],
          "target-port-range": [
              {
                "lower-port": number,
                "upper-port": number
              }
           ],
           "target-protocol": [
             number
           ],
           "target-fqdn": [
             "string"
           ],
           "target-uri": [
             "string"
           ],
           "alias-name": [
             "string"
           ],
          "lifetime": number,
          "trigger-mitigation": true|false
        }
      ]
    }
  }
```

            Figure 6: PUT to Convey DOTS Mitigation Requests (Message Body
                                    Schema)

   The parameters in the CBOR body (Figure 6) of the PUT request are
   described below:

   target-prefix: A list of prefixes identifying resources under
      attack. Prefixes are represented using Classless Inter-Domain
      Routing (CIDR) notation [RFC4632].
      As a reminder, the prefix length must be less than or equal to 32
      (or 128) for IPv4 (or IPv6).

      The prefix list MUST NOT include broadcast, loopback, or multicast
      addresses. These addresses are considered as invalid values. In
      addition, the DOTS server MUST validate that target prefixes are

within the scope of the DOTS client domain. Other validation
checks may be supported by DOTS servers.

This is an optional attribute.

target-port-range: A list of port numbers bound to resources under
attack.

A port range is defined by two bounds, a lower port number (lower-
port) and an upper port number (upper-port). When only 'lower-
port' is present, it represents a single port number.

For TCP, UDP, Stream Control Transmission Protocol (SCTP)
[RFC4960], or Datagram Congestion Control Protocol (DCCP)
[RFC4340], a range of ports can be, for example, 0-1023,
1024-65535, or 1024-49151.

This is an optional attribute.

target-protocol: A list of protocols involved in an attack. Values
are taken from the IANA protocol registry [proto_numbers].

If 'target-protocol' is not specified, then the request applies to
any protocol.

This is an optional attribute.

target-fqdn: A list of Fully Qualified Domain Names (FQDNs)
identifying resources under attack [RFC8499].

How a name is passed to an underlying name resolution library is
implementation- and deployment-specific. Nevertheless, once the
name is resolved into one or multiple IP addresses, DOTS servers
MUST apply the same validation checks as those for 'target-
prefix'.

The use of FQDNs may be suboptimal because:

* It induces both an extra load and increased delays on the DOTS
   server to handle and manage DNS resolution requests.

* It does not guarantee that the DOTS server will resolve a name
   to the same IP addresses that the DOTS client does.

This is an optional attribute.

target-uri: A list of Uniform Resource Identifiers (URIs) [RFC3986]
identifying resources under attack.

The same validation checks used for 'target-fqdn' MUST be followed by DOTS servers to validate a target URI.

This is an optional attribute.

alias-name: A list of aliases of resources for which the mitigation is requested. Aliases can be created using the DOTS data channel (Section 6.1 of [I-D.ietf-dots-data-channel]), direct configuration, or other means.

An alias is used in subsequent signal channel exchanges to refer more efficiently to the resources under attack.

This is an optional attribute.

lifetime: Lifetime of the mitigation request in seconds. The RECOMMENDED lifetime of a mitigation request is 3600 seconds -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while still making the request expire in a timely manner when the client has unexpectedly quit. DOTS clients MUST include this parameter in their mitigation requests. Upon the expiry of this lifetime, and if the request is not refreshed, the mitigation request is removed. The request can be refreshed by sending the same request again.

A lifetime of '0' in a mitigation request is an invalid value.

A lifetime of negative one (-1) indicates indefinite lifetime for the mitigation request. The DOTS server MAY refuse indefinite lifetime, for policy reasons; the granted lifetime value is returned in the response. DOTS clients MUST be prepared to not be granted mitigations with indefinite lifetimes.

The DOTS server MUST always indicate the actual lifetime in the response and the remaining lifetime in status messages sent to the DOTS client.

This is a mandatory attribute.

trigger-mitigation: If the parameter value is set to 'false', DDoS mitigation will not be triggered for the mitigation request unless the DOTS signal channel session is lost.

If the DOTS client ceases to respond to heartbeat messages, the DOTS server can detect that the DOTS signal channel session is lost. More details are discussed in Section 4.7.

      The default value of the parameter is 'true' (that is, the
      mitigation starts immediately). If 'trigger-mitigation' is not
      present in a request, this is equivalent to receiving a request
      with 'trigger-mitigation' set to 'true'.

      This is an optional attribute.

   In deployments where server-domain DOTS gateways are enabled,
   identity information about the origin source client domain ('cdid')
   SHOULD be propagated to the DOTS server. That information is meant
   to assist the DOTS server to enforce some policies such as grouping
   DOTS clients that belong to the same DOTS domain, limiting the number
   of DOTS requests, and identifying the mitigation scope. These
   policies can be enforced per-client, per-client domain, or both.
   Also, the identity information may be used for auditing and debugging
   purposes.

   Figure 7 shows an example of a request relayed by a server-domain
   DOTS gateway.

```
  Header: PUT (Code=0.03)
  Uri-Path: ".well-known"
  Uri-Path: "dots"
  Uri-Path: "mitigate"
  Uri-Path: "cdid=7eeaf349529eb55ed50113"
  Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
  Uri-Path: "mid=123"
  Content-Format: "application/dots+cbor"

  {
    ...
  }
```

        Figure 7: PUT for DOTS Mitigation Request as Relayed by a DOTS
                                  Gateway

   A server-domain DOTS gateway SHOULD add the following Uri-Path
   parameter:

   cdid: Stands for Client Domain Identifier. The 'cdid' is conveyed by
         a server-domain DOTS gateway to propagate the source domain
         identity from the gateway's client-facing-side to the gateway's
         server-facing-side, and from the gateway's server-facing-side
         to the DOTS server. 'cdid' may be used by the final DOTS server
         for policy enforcement purposes (e.g., enforce a quota on
         filtering rules). These policies are deployment-specific.

Server-domain DOTS gateways SHOULD support a configuration
option to instruct whether 'cdid' parameter is to be inserted.

In order to accommodate deployments that require enforcing per-
client policies, per-client domain policies, or a combination
thereof, server-domain DOTS gateways instructed to insert the
'cdid' parameter MUST supply the SPKI hash of the DOTS client
X.509 certificate, the DOTS client raw public key, or the hash
of the "PSK identity" in the 'cdid', following the same rules
for generating the hash conveyed in 'cuid', which is then used
by the ultimate DOTS server to determine the corresponding
client's domain. The 'cdid' generated by a server-domain
gateway is likely to be the same as the 'cuid' except if the
DOTS message was relayed by a client-domain DOTS gateway or the
'cuid' was generated from a rogue DOTS client.

If a DOTS client is provisioned, for example, with distinct
certificates as a function of the peer server-domain DOTS
gateway, distinct 'cdid' values may be supplied by a server-
domain DOTS gateway. The ultimate DOTS server MUST treat those
'cdid' values as equivalent.

The 'cdid' attribute MUST NOT be generated and included by DOTS
clients.

DOTS servers MUST ignore 'cdid' attributes that are directly
supplied by source DOTS clients or client-domain DOTS gateways.
This implies that first server-domain DOTS gateways MUST strip
'cdid' attributes supplied by DOTS clients. DOTS servers
SHOULD support a configuration parameter to identify DOTS
gateways that are trusted to supply 'cdid' attributes.

Only single-valued 'cdid' are defined in this document. That
is, only the first on-path server-domain DOTS gateway can
insert a 'cdid' value. This specification does not allow
multiple server-domain DOTS gateways, whenever involved in the
path, to insert a 'cdid' value for each server-domain gateway.

This is an optional Uri-Path. When present, 'cdid' MUST be
positioned before 'cuid'.

A DOTS gateway MAY add the CoAP Hop-Limit Option
[I-D.ietf-core-hop-limit].

Because of the complexity to handle partial failure cases, this
specification does not allow for including multiple mitigation
requests in the same PUT request. Concretely, a DOTS client MUST NOT

include multiple entries in the 'scope' array of the same PUT
request.

FQDN and URI mitigation scopes may be thought of as a form of scope
alias, in which the addresses associated with the domain name or URI
(as resolved by the DOTS server) represent the scope of the
mitigation. Particularly, the IP addresses to which the host
subcomponent of authority component of an URI resolves represent the
'target-prefix', the URI scheme represents the 'target-protocol', the
port subcomponent of authority component of an URI represents the
'target-port-range'. If the optional port information is not present
in the authority component, the default port defined for the URI
scheme represents the 'target-port'.

In the PUT request at least one of the attributes 'target-prefix',
'target-fqdn','target-uri', or 'alias-name' MUST be present.

Attributes and Uri-Path parameters with empty values MUST NOT be
present in a request and render the entire request invalid.

Figure 8 shows a PUT request example to signal that TCP port numbers
80, 8080, and 443 used by 2001:db8:6401::1 and 2001:db8:6401::2
servers are under attack. The presence of 'cdid' indicates that a
server-domain DOTS gateway has modified the initial PUT request sent
by the DOTS client. Note that 'cdid' MUST NOT appear in the PUT
request message body.

```
    Header: PUT (Code=0.03)
    Uri-Path: ".well-known"
    Uri-Path: "dots"
    Uri-Path: "mitigate"
    Uri-Path: "cdid=7eeaf349529eb55ed50113"
    Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
    Uri-Path: "mid=123"
    Content-Format: "application/dots+cbor"

    {
      "ietf-dots-signal-channel:mitigation-scope": {
        "scope": [
          {
            "target-prefix": [
               "2001:db8:6401::1/128",
               "2001:db8:6401::2/128"
             ],
            "target-port-range": [
              {
                "lower-port": 80
              },
              {
                "lower-port": 443
              },
              {
                 "lower-port": 8080
              }
             ],
             "target-protocol": [
               6
             ],
            "lifetime": 3600
          }
        ]
      }
    }
```

                Figure 8: PUT for DOTS Mitigation Request (An Example)

   The corresponding CBOR encoding format for the payload is shown in
   Figure 9.

```
  A1                                         # map(1)
     01                                      # unsigned(1)
     A1                                      # map(1)
        02                                   # unsigned(2)
        81                                   # array(1)
           A3                                # map(3)
              06                             # unsigned(6)
              82                             # array(2)
                 74                          # text(20)
                    323030313A6462383A363430313A3A312F313238
                 74                          # text(20)
                    323030313A6462383A363430313A3A322F313238
              07                             # unsigned(7)
              83                             # array(3)
              A1                             # map(1)
                 08                          # unsigned(8)
                 18 50                       # unsigned(80)
              A1                             # map(1)
                 08                          # unsigned(8)
                 19 01BB                     # unsigned(443)
              A1                             # map(1)
                 08                          # unsigned(8)
                 19 1F90                     # unsigned(8080)
           0A                                # unsigned(10)
           81                                # array(1)
              06                             # unsigned(6)
           0E                                # unsigned(14)
              19 0E10                        # unsigned(3600)
```

                 Figure 9: PUT for DOTS Mitigation Request (CBOR)

   In both DOTS signal and data channel sessions, the DOTS client MUST
   authenticate itself to the DOTS server (Section 8). The DOTS server
   MAY use the algorithm presented in Section 7 of [RFC7589] to derive
   the DOTS client identity or username from the client certificate.
   The DOTS client identity allows the DOTS server to accept mitigation
   requests with scopes that the DOTS client is authorized to manage.

   The DOTS server couples the DOTS signal and data channel sessions
   using the DOTS client identity and optionally the 'cdid' parameter
   value, so the DOTS server can validate whether the aliases conveyed
   in the mitigation request were indeed created by the same DOTS client
   using the DOTS data channel session. If the aliases were not created
   by the DOTS client, the DOTS server MUST return 4.00 (Bad Request) in
   the response.

   The DOTS server couples the DOTS signal channel sessions using the
   DOTS client identity and optionally the 'cdid' parameter value, and

the DOTS server uses 'mid' and 'cuid' Uri-Path parameter values to
detect duplicate mitigation requests. If the mitigation request
contains the 'alias-name' and other parameters identifying the target
resources (such as 'target-prefix', 'target-port-range', 'target-
fqdn', or 'target-uri'), the DOTS server appends the parameter values
in 'alias-name' with the corresponding parameter values in 'target-
prefix', 'target-port-range', 'target-fqdn', or 'target-uri'.

The DOTS server indicates the result of processing the PUT request
using CoAP response codes. CoAP 2.xx codes are success. CoAP 4.xx
codes are some sort of invalid requests (client errors). COAP 5.xx
codes are returned if the DOTS server is in an error state or is
currently unavailable to provide mitigation in response to the
mitigation request from the DOTS client.

Figure 10 shows an example response to a PUT request that is
successfully processed by a DOTS server (i.e., CoAP 2.xx response
codes). This version of the specification forbids 'cuid' and 'cdid'
(if used) to be returned in a response message body.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 123,
        "lifetime": 3600
      }
    ]
  }
}
```

                    Figure 10: 2.xx Response Body

If the request is missing a mandatory attribute, does not include
'cuid' or 'mid' Uri-Path options, includes multiple 'scope'
parameters, or contains invalid or unknown parameters, the DOTS
server MUST reply with 4.00 (Bad Request). DOTS agents can safely
ignore comprehension-optional parameters they don't understand
(Section 9.6.1.1).

A DOTS server that receives a mitigation request with a lifetime set
to '0' MUST reply with a 4.00 (Bad Request).

If the DOTS server does not find the 'mid' parameter value conveyed
in the PUT request in its configuration data, it MAY accept the
mitigation request by sending back a 2.01 (Created) response to the
DOTS client; the DOTS server will consequently try to mitigate the
attack. A DOTS server could reject mitigation requests when it is

near capacity or needs to rate-limit a particular client, for
example.

The relative order of two mitigation requests, having the same
'trigger-mitigation' type, from a DOTS client is determined by
comparing their respective 'mid' values. If two mitigation requests
with the same 'trigger-mitigation' type have overlapping mitigation
scopes, the mitigation request with the highest numeric 'mid' value
will override the other mitigation request. Two mitigation requests
from a DOTS client have overlapping scopes if there is a common IP
address, IP prefix, FQDN, URI, or alias-name. To avoid maintaining a
long list of overlapping mitigation requests (i.e., requests with the
same 'trigger-mitigation' type and overlapping scopes) from a DOTS
client and avoid error-prone provisioning of mitigation requests from
a DOTS client, the overlapped lower numeric 'mid' MUST be
automatically deleted and no longer available at the DOTS server.
For example, if the DOTS server receives a mitigation request which
overlaps with an existing mitigation with a higher numeric 'mid', the
DOTS server rejects the request by returning 4.09 (Conflict) to the
DOTS client. The response includes enough information for a DOTS
client to recognize the source of the conflict as described below in
the 'conflict-information' subtree with only the relevant nodes
listed:

conflict-information: Indicates that a mitigation request is
   conflicting with another mitigation request. This optional
   attribute has the following structure:

   conflict-cause: Indicates the cause of the conflict. The
      following values are defined:

      1: Overlapping targets. 'conflict-scope' provides more details
          about the conflicting target clauses.

   conflict-scope: Characterizes the exact conflict scope. It may
      include a list of IP addresses, a list of prefixes, a list of
      port numbers, a list of target protocols, a list of FQDNs, a
      list of URIs, a list of alias-names, or a 'mid'.

If the DOTS server receives a mitigation request which overlaps with
an active mitigation request, but both having distinct 'trigger-
mitigation' types, the DOTS server SHOULD deactivate (absent explicit
policy/configuration otherwise) the mitigation request with 'trigger-
mitigation' set to false. Particularly, if the mitigation request
with 'trigger-mitigation' set to false is active, the DOTS server
withdraws the mitigation request (i.e., status code is set to '7' as
defined in Table 2) and transitions the status of the mitigation
request to '8'.

Upon DOTS signal channel session loss with a peer DOTS client, the
DOTS server SHOULD withdraw (absent explicit policy/configuration
otherwise) any active mitigation requests overlapping with mitigation
requests having 'trigger-mitigation' set to false from that DOTS
client, as the loss of the session implicitly activates these
preconfigured mitigation requests and they take precedence. Note
that active-but-terminating period is not observed for mitigations
withdrawn at the initiative of the DOTS server.

DOTS clients may adopt various strategies for setting the scopes of
immediate and pre-configured mitigation requests to avoid potential
conflicts. For example, a DOTS client may tweak pre-configured
scopes so that the scope of any overlapping immediate mitigation
request will be a subset of the pre-configured scopes. Also, if an
immediate mitigation request overlaps with any of the pre-configured
scopes, the DOTS client sets the scope of the overlapping immediate
mitigation request to be a subset of the pre-configured scopes, so as
to get a broad mitigation when the DOTS signal channel collapses and
maximize the chance of recovery.

If the request is conflicting with an existing mitigation request
from a different DOTS client, the DOTS server may return 2.01
(Created) or 4.09 (Conflict) to the requesting DOTS client. If the
DOTS server decides to maintain the new mitigation request, the DOTS
server returns 2.01 (Created) to the requesting DOTS client. If the
DOTS server decides to reject the new mitigation request, the DOTS
server returns 4.09 (Conflict) to the requesting DOTS client. For
both 2.01 (Created) and 4.09 (Conflict) responses, the response
includes enough information for a DOTS client to recognize the source
of the conflict as described below:

conflict-information: Indicates that a mitigation request is
   conflicting with another mitigation request(s) from other DOTS
   client(s). This optional attribute has the following structure:

   conflict-status: Indicates the status of a conflicting mitigation
      request. The following values are defined:

      1: DOTS server has detected conflicting mitigation requests
          from different DOTS clients. This mitigation request is
          currently inactive until the conflicts are resolved.
          Another mitigation request is active.

      2: DOTS server has detected conflicting mitigation requests
          from different DOTS clients. This mitigation request is
          currently active.

3: DOTS server has detected conflicting mitigation requests
    from different DOTS clients. All conflicting mitigation
    requests are inactive.

conflict-cause: Indicates the cause of the conflict. The
   following values are defined:

   1: Overlapping targets. 'conflict-scope' provides more details
       about the conflicting target clauses.

   2: Conflicts with an existing accept-list. This code is
       returned when the DDoS mitigation detects source addresses/
       prefixes in the accept-listed ACLs are attacking the
       target.

   3: CUID Collision. This code is returned when a DOTS client
       uses a 'cuid' that is already used by another DOTS client.
       This code is an indication that the request has been
       rejected and a new request with a new 'cuid' is to be re-
       sent by the DOTS client (see the example shown in
       Figure 11). Note that 'conflict-status', 'conflict-scope',
       and 'retry-timer' MUST NOT be returned in the error
       response.

conflict-scope: Characterizes the exact conflict scope. It may
   include a list of IP addresses, a list of prefixes, a list of
   port numbers, a list of target protocols, a list of FQDNs, a
   list of URIs, a list of alias-names, or references to
   conflicting ACLs (by an 'acl-name', typically
   [I-D.ietf-dots-data-channel]).

retry-timer: Indicates, in seconds, the time after which the DOTS
   client may re-issue the same request. The DOTS server returns
   'retry-timer' only to DOTS client(s) for which a mitigation
   request is deactivated. Any retransmission of the same
   mitigation request before the expiry of this timer is likely to
   be rejected by the DOTS server for the same reasons.

   The retry-timer SHOULD be equal to the lifetime of the active
   mitigation request resulting in the deactivation of the
   conflicting mitigation request.

   If the DOTS server decides to maintain a state for the
   deactivated mitigation request, the DOTS server updates the
   lifetime of the deactivated mitigation request to 'retry-timer
   + 45 seconds' (that is, this mitigation request remains
   deactivated for the entire duration of 'retry-timer + 45
   seconds') so that the DOTS client can refresh the deactivated

mitigation request after 'retry-timer' seconds, but before the
expiry of the lifetime, and check if the conflict is resolved.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=7eeaf349529eb55ed50113"
Uri-Path: "mid=12"

(1) Request with a conflicting 'cuid'

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "conflict-information": {
          "conflict-cause": "cuid-collision"
        }
      }
    ]
  }
}

(2) Message body of the 4.09 (Conflict) response
  from the DOTS server

Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=f30d281ce6b64fc5a0b91e"
Uri-Path: "mid=12"

(3) Request with a new 'cuid'
```

Figure 11: Example of Generating a New 'cuid'

As an active attack evolves, DOTS clients can adjust the scope of
requested mitigation as necessary, by refining the scope of resources
requiring mitigation. This can be achieved by sending a PUT request
with a new 'mid' value that will override the existing one with
overlapping mitigation scopes.

For a mitigation request to continue beyond the initial negotiated
lifetime, the DOTS client has to refresh the current mitigation
request by sending a new PUT request. This PUT request MUST use the
same 'mid' value, and MUST repeat all the other parameters as sent in

the original mitigation request apart from a possible change to the
lifetime parameter value. In such case, the DOTS server MAY update
the mitigation request, and a 2.04 (Changed) response is returned to
indicate a successful update of the mitigation request. If this is
not the case, the DOTS server MUST reject the request with a 4.00
(Bad Request).

4.4.2. Retrieve Information Related to a Mitigation

   A GET request is used by a DOTS client to retrieve information
   (including status) of DOTS mitigations from a DOTS server.

   'cuid' is a mandatory Uri-Path parameter for GET requests.

   Uri-Path parameters with empty values MUST NOT be present in a
   request.

   The same considerations for manipulating 'cdid' parameter by server-
   domain DOTS gateways specified in Section 4.4.1 MUST be followed for
   GET requests.

   The 'c' Uri-Query option is used to control selection of
   configuration and non-configuration data nodes. Concretely, the 'c'
   (content) parameter and its permitted values defined in the following
   table [I-D.ietf-core-comi] can be used to retrieve non-configuration
   data (attack mitigation status), configuration data, or both. The
   DOTS server MAY support this optional filtering capability. It can
   safely ignore it if not supported. If the DOTS client supports the
   optional filtering capability, it SHOULD use "c=n" query (to get back
   only the dynamically changing data) or "c=c" query (to get back the
   static configuration values) when the DDoS attack is active to limit
   the size of the response.

```
       +-------+-----------------------------------------------------+
       | Value | Description                                         |
       +-------+-----------------------------------------------------+
       |   c   | Return only configuration descendant data nodes     |
       |   n   | Return only non-configuration descendant data nodes |
       |   a   | Return all descendant data nodes                    |
       +-------+-----------------------------------------------------+
```

   The DOTS client can use Block-wise transfer [RFC7959] to get the list
   of all its mitigations maintained by a DOTS server, it can send
   Block2 Option in a GET request with NUM = 0 to aid in limiting the
   size of the response. If the representation of all the active
   mitigation requests associated with the DOTS client does not fit
   within a single datagram, the DOTS server MUST use the Block2 Option
   with NUM = 0 in the GET response. The Size2 Option may be conveyed

in the response to indicate the total size of the resource
representation. The DOTS client retrieves the rest of the
representation by sending additional GET requests with Block2 Options
containing NUM values greater than zero. The DOTS client MUST adhere
to the block size preferences indicated by the DOTS server in the
response. If the DOTS server uses the Block2 Option in the GET
response and the response is for a dynamically changing resource
(e.g., "c=n" or "c=a" query), the DOTS server MUST include the ETag
Option in the response. The DOTS client MUST include the same ETag
value in subsequent GET requests to retrieve the rest of the
representation.

The following examples illustrate how a DOTS client retrieves active
mitigation requests from a DOTS server. In particular:

o Figure 12 shows the example of a GET request to retrieve all DOTS
   mitigation requests signaled by a DOTS client.

o Figure 13 shows the example of a GET request to retrieve a
   specific DOTS mitigation request signaled by a DOTS client. The
   configuration data to be reported in the response is formatted in
   the same order as was processed by the DOTS server in the original
   mitigation request.

These two examples assume the default of "c=a"; that is, the DOTS
client asks for all data to be reported by the DOTS server.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Observe: 0
```

       Figure 12: GET to Retrieve all DOTS Mitigation Requests

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=12332"
Observe: 0
```

      Figure 13: GET to Retrieve a Specific DOTS Mitigation Request

If the DOTS server does not find the 'mid' Uri-Path value conveyed in
the GET request in its configuration data for the requesting DOTS

client, it MUST respond with a 4.04 (Not Found) error response code.
Likewise, the same error MUST be returned as a response to a request
to retrieve all mitigation records (i.e., 'mid' Uri-Path is not
defined) of a given DOTS client if the DOTS server does not find any
mitigation record for that DOTS client. As a reminder, a DOTS client
is identified by its identity (e.g., client certificate, 'cuid') and
optionally the 'cdid'.

Figure 14 shows a response example of all active mitigation requests
associated with the DOTS client as maintained by the DOTS server.
The response indicates the mitigation status of each mitigation
request.

```
   {
     "ietf-dots-signal-channel:mitigation-scope": {
       "scope": [
         {
           "mid": 12332,
           "mitigation-start": "1507818434",
           "target-prefix": [
               "2001:db8:6401::1/128",
               "2001:db8:6401::2/128"
           ],
           "target-protocol": [
             17
           ],
           "lifetime": 1756,
           "status": "attack-successfully-mitigated",
           "bytes-dropped": "134334555",
           "bps-dropped": "43344",
           "pkts-dropped": "333334444",
           "pps-dropped": "432432"
         },
         {
           "mid": 12333,
           "mitigation-start": "1507818393",
           "target-prefix": [
               "2001:db8:6401::1/128",
               "2001:db8:6401::2/128"
           ],
           "target-protocol": [
             6
           ],
           "lifetime": 1755,
           "status": "attack-stopped",
           "bytes-dropped": "0",
           "bps-dropped": "0",
           "pkts-dropped": "0",
           "pps-dropped": "0"
         }
       ]
     }
   }
```

                  Figure 14: Response Body to a GET Request

   The mitigation status parameters are described below:

   mitigation-start: Mitigation start time is expressed in seconds
      relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of

   [RFC7049]). The CBOR encoding is modified so that the leading tag
   1 (epoch-based date/time) MUST be omitted.

   This is a mandatory attribute when an attack mitigation is active.
   Particularly, 'mitigation-start' is not returned for a mitigation
   with 'status' code set to 8.

   lifetime: The remaining lifetime of the mitigation request, in
      seconds.

      This is a mandatory attribute.

   status: Status of attack mitigation. The various possible values of
      'status' parameter are explained in Table 2.

      This is a mandatory attribute.

   bytes-dropped: The total dropped byte count for the mitigation
      request since the attack mitigation is triggered. The count wraps
      around when it reaches the maximum value of unsigned integer64.

      This is an optional attribute.

   bps-dropped: The average number of dropped bytes per second for the
      mitigation request since the attack mitigation is triggered. This
      average SHOULD be over five-minute intervals (that is, measuring
      bytes into five-minute buckets and then averaging these buckets
      over the time since the mitigation was triggered).

      This is an optional attribute.

   pkts-dropped: The total number of dropped packet count for the
      mitigation request since the attack mitigation is triggered. The
      count wraps around when it reaches the maximum value of unsigned
      integer64.

      This is an optional attribute.

   pps-dropped: The average number of dropped packets per second for
      the mitigation request since the attack mitigation is triggered.
      This average SHOULD be over five-minute intervals (that is,
      measuring packets into five-minute buckets and then averaging
      these buckets over the time since the mitigation was triggered).

      This is an optional attribute.

+-----------+--+
| Parameter | Description |
| Value | |
+-----------+--+
1	Attack mitigation setup is in progress (e.g.,
	changing the network path to redirect the inbound
	traffic to a DOTS mitigator).
+-----------+--+
2	Attack is being successfully mitigated (e.g., traffic
	is redirected to a DDoS mitigator and attack traffic
	is dropped).
+-----------+--+
3	Attack has stopped and the DOTS client can withdraw
	the mitigation request. This status code will be
	transmitted for immediate mitigation requests till
	the mitigation is withdrawn or the lifetime expires.
	For mitigation requests with pre-configured scopes
	(i.e., 'trigger-mitigation' set to 'false'), this
	status code will be transmitted 4 times and then
	transition to "8".
+-----------+--+
| 4 | Attack has exceeded the mitigation provider |
| | capability. |
+-----------+--+
| 5 | DOTS client has withdrawn the mitigation request and |
| | the mitigation is active but terminating. |
+-----------+--+
| 6 | Attack mitigation is now terminated. |
+-----------+--+
7	Attack mitigation is withdrawn (by the DOTS server).
	If a mitigation request with 'trigger-mitigation' set
	to false is withdrawn because it overlaps with an
	immediate mitigation request, this status code will
	be transmitted 4 times and then transition to "8" for
	the mitigation request with pre-configured scopes.
+-----------+--+
8	Attack mitigation will be triggered for the
	mitigation request only when the DOTS signal channel
	session is lost.
+-----------+--+

                    Table 2: Values of 'status' Parameter

4.4.2.1. DOTS Servers Sending Mitigation Status

   The Observe Option defined in [RFC7641] extends the CoAP core
   protocol with a mechanism for a CoAP client to "observe" a resource
   on a CoAP server: The client retrieves a representation of the

resource and requests this representation be updated by the server as long as the client is interested in the resource. DOTS implementations MUST use the Observe Option for both 'mitigate' and 'config' (Section 4.2).

A DOTS client conveys the Observe Option set to '0' in the GET request to receive asynchronous notifications of attack mitigation status from the DOTS server.

Unidirectional mitigation notifications within the bidirectional signal channel enables asynchronous notifications between the agents. [RFC7641] indicates that (1) a notification can be sent in a Confirmable or a Non-confirmable message, and (2) the message type used is typically application-dependent and may be determined by the server for each notification individually. For DOTS server application, the message type MUST always be set to Non-confirmable even if the underlying COAP library elects a notification to be sent in a Confirmable message. This overrides the behavior defined in Section 4.5 of [RFC7641] to send a Confirmable message instead of a Non-confirmable message at least every 24 hour for the following reasons: First, the DOTS signal channel uses a heartbeat mechanism to determine if the DOTS client is alive. Second, Confirmable messages are not suitable during an attack.

Due to the higher likelihood of packet loss during a DDoS attack, the DOTS server periodically sends attack mitigation status to the DOTS client and also notifies the DOTS client whenever the status of the attack mitigation changes. If the DOTS server cannot maintain an RTT estimate, it MUST NOT send more than one asynchronous notification every 3 seconds, and SHOULD use an even less aggressive rate whenever possible (case 2 in Section 3.1.3 of [RFC8085]).

When conflicting requests are detected, the DOTS server enforces the corresponding policy (e.g., accept all requests, reject all requests, accept only one request but reject all the others, ...). It is assumed that this policy is supplied by the DOTS server administrator or it is a default behavior of the DOTS server implementation. Then, the DOTS server sends notification message(s) to the DOTS client(s) at the origin of the conflict (refer to the conflict parameters defined in Section 4.4.1). A conflict notification message includes information about the conflict cause, scope, and the status of the mitigation request(s). For example,

o A notification message with 'status' code set to '7 (Attack mitigation is withdrawn)' and 'conflict-status' set to '1' is sent to a DOTS client to indicate that an active mitigation request is deactivated because a conflict is detected.

o A notification message with 'status' code set to '1 (Attack
   mitigation is in progress)' and 'conflict-status' set to '2' is
   sent to a DOTS client to indicate that this mitigation request is
   in progress, but a conflict is detected.

Upon receipt of a conflict notification message indicating that a
mitigation request is deactivated because of a conflict, a DOTS
client MUST NOT resend the same mitigation request before the expiry
of 'retry-timer'. It is also recommended that DOTS clients support
means to alert administrators about mitigation conflicts.

A DOTS client that is no longer interested in receiving notifications
from the DOTS server can simply "forget" the observation. When the
DOTS server sends the next notification, the DOTS client will not
recognize the token in the message and thus will return a Reset
message. This causes the DOTS server to remove the associated entry.
Alternatively, the DOTS client can explicitly deregister itself by
issuing a GET request that has the Token field set to the token of
the observation to be cancelled and includes an Observe Option with
the value set to '1' (deregister). The latter is RECOMMENDED.

Figure 15 shows an example of a DOTS client requesting a DOTS server
to send notifications related to a mitigation request. Note that for
mitigations with pre-configured scopes (i.e., 'trigger-mitigation'
set to 'false'), the state will need to transition from 3 (attack-
stopped) to 8 (attack-mitigation-signal-loss).

```
    +-----------+                           +-----------+
    |DOTS client|                           |DOTS server|
    +-----------+                           +-----------+
        |                                       |
        |    GET /<mid>                         |
        |    Token: 0x4a                        | Registration
        |    Observe: 0                         |
        +-------------------------------------->|
        |                                       |
        |    2.05 Content                       |
        |    Token: 0x4a                        | Notification of
        |    Observe: 12                        | the current state
        |    status: "attack-mitigation-in-progress"
        |                                       |
        |<--------------------------------------+
        |    2.05 Content                       |
        |    Token: 0x4a                        | Notification upon
        |    Observe: 44                        | a state change
        |    status: "attack-successfully-mitigated"
        |                                       |
        |<--------------------------------------+
        |    2.05 Content                       |
        |    Token: 0x4a                        | Notification upon
        |    Observe: 60                        | a state change
        |    status: "attack-stopped"           |
        |<--------------------------------------+
        |                                       |
                        ...
```

Figure 15: Notifications of Attack Mitigation Status

4.4.2.2. DOTS Clients Polling for Mitigation Status

   The DOTS client can send the GET request at frequent intervals
   without the Observe Option to retrieve the configuration data of the
   mitigation request and non-configuration data (i.e., the attack
   status). DOTS clients MAY be configured with a policy indicating the
   frequency of polling DOTS servers to get the mitigation status. This
   frequency MUST NOT be more than one UDP datagram per RTT as discussed
   in Section 3.1.3 of [RFC8085].

   If the DOTS server has been able to mitigate the attack and the
   attack has stopped, the DOTS server indicates as such in the status.
   In such case, the DOTS client recalls the mitigation request by
   issuing a DELETE request for this mitigation request (Section 4.4.4).

   A DOTS client SHOULD react to the status of the attack as per the
   information sent by the DOTS server rather than performing its own

detection that the attack has been mitigated. This ensures that the
DOTS client does not recall a mitigation request prematurely because
it is possible that the DOTS client does not sense the DDoS attack on
its resources, but the DOTS server could be actively mitigating the
attack because the attack is not completely averted.

4.4.3. Efficacy Update from DOTS Clients

   While DDoS mitigation is in progress, due to the likelihood of packet
   loss, a DOTS client MAY periodically transmit DOTS mitigation
   efficacy updates to the relevant DOTS server. A PUT request is used
   to convey the mitigation efficacy update to the DOTS server. This
   PUT request is treated as a refresh of the current mitigation.

   The PUT request used for efficacy update MUST include all the
   parameters used in the PUT request to carry the DOTS mitigation
   request (Section 4.4.1) unchanged apart from the 'lifetime' parameter
   value. If this is not the case, the DOTS server MUST reject the
   request with a 4.00 (Bad Request).

   The If-Match Option (Section 5.10.8.1 of [RFC7252]) with an empty
   value is used to make the PUT request conditional on the current
   existence of the mitigation request. If UDP is used as transport,
   CoAP requests may arrive out-of-order. For example, the DOTS client
   may send a PUT request to convey an efficacy update to the DOTS
   server followed by a DELETE request to withdraw the mitigation
   request, but the DELETE request arrives at the DOTS server before the
   PUT request. To handle out-of-order delivery of requests, if an If-
   Match Option is present in the PUT request and the 'mid' in the
   request matches a mitigation request from that DOTS client, the
   request is processed by the DOTS server. If no match is found, the
   PUT request is silently ignored by the DOTS server.

   An example of an efficacy update message, which includes an If-Match
   Option with an empty value, is depicted in Figure 16.

```
     Header: PUT (Code=0.03)
     Uri-Path: ".well-known"
     Uri-Path: "dots"
     Uri-Path: "mitigate"
     Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
     Uri-Path: "mid=123"
     If-Match:
     Content-Format: "application/dots+cbor"

     {
      "ietf-dots-signal-channel:mitigation-scope": {
        "scope": [
          {
            "target-prefix": [
              "2001:db8:6401::1/128",
              "2001:db8:6401::2/128"
            ],
            "target-port-range": [
              {
                "lower-port": 80
              },
              {
                "lower-port": 443
              },
              {
                 "lower-port": 8080
              }
            ],
            "target-protocol": [
               6
            ],
            "attack-status": "under-attack"
          }
        ]
      }
     }
```

               Figure 16: An Example of Efficacy Update

   The 'attack-status' parameter is a mandatory attribute when
   performing an efficacy update. The various possible values contained
   in the 'attack-status' parameter are described in Table 3.

```
+-----------+-------------------------------------------------------+
| Parameter | Description                                           |
|     value |                                                       |
+-----------+-------------------------------------------------------+
|         1 | The DOTS client determines that it is still under     |
|           | attack.                                               |
+-----------+-------------------------------------------------------+
|         2 | The DOTS client determines that the attack is         |
|           | successfully mitigated (e.g., attack traffic is not   |
|           | seen).                                                |
+-----------+-------------------------------------------------------+
```

                Table 3: Values of 'attack-status' Parameter

   The DOTS server indicates the result of processing a PUT request
   using CoAP response codes. The response code 2.04 (Changed) is
   returned if the DOTS server has accepted the mitigation efficacy
   update. The error response code 5.03 (Service Unavailable) is
   returned if the DOTS server has erred or is incapable of performing
   the mitigation. As specified in [RFC7252], 5.03 uses Max-Age option
   to indicate the number of seconds after which to retry.

4.4.4. Withdraw a Mitigation

   DELETE requests are used to withdraw DOTS mitigation requests from
   DOTS servers (Figure 17).

   'cuid' and 'mid' are mandatory Uri-Path parameters for DELETE
   requests.

   The same considerations for manipulating 'cdid' parameter by DOTS
   gateways, as specified in Section 4.4.1, MUST be followed for DELETE
   requests. Uri-Path parameters with empty values MUST NOT be present
   in a request.

```
   Header: DELETE (Code=0.04)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "mitigate"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "mid=123"
```

                   Figure 17: Withdraw a DOTS Mitigation

   If the DELETE request does not include 'cuid' and 'mid' parameters,
   the DOTS server MUST reply with a 4.00 (Bad Request).

Once the request is validated, the DOTS server immediately acknowledges a DOTS client's request to withdraw the DOTS signal using 2.02 (Deleted) response code with no response payload. A 2.02 (Deleted) Response Code is returned even if the 'mid' parameter value conveyed in the DELETE request does not exist in its configuration data before the request.

If the DOTS server finds the 'mid' parameter value conveyed in the DELETE request in its configuration data for the DOTS client, then to protect against route or DNS flapping caused by a DOTS client rapidly removing a mitigation, and to dampen the effect of oscillating attacks, the DOTS server MAY allow mitigation to continue for a limited period after acknowledging a DOTS client's withdrawal of a mitigation request. During this period, the DOTS server status messages SHOULD indicate that mitigation is active but terminating (Section 4.4.2).

The initial active-but-terminating period SHOULD be sufficiently long to absorb latency incurred by route propagation. The active-but-terminating period SHOULD be set by default to 120 seconds. If the client requests mitigation again before the initial active-but-terminating period elapses, the DOTS server MAY exponentially increase (the base of the exponent is 2) the active-but-terminating period up to a maximum of 300 seconds (5 minutes).

Once the active-but-terminating period elapses, the DOTS server MUST treat the mitigation as terminated, as the DOTS client is no longer responsible for the mitigation.

If a mitigation is triggered due to a signal channel loss, the DOTS server relies upon normal triggers to stop that mitigation (typically, receipt of a valid DELETE request, expiry of the mitigation lifetime, or scrubbing the traffic to the attack target). In particular, the DOTS server MUST NOT consider the signal channel recovery as a trigger to stop the mitigation.

4.5. DOTS Signal Channel Session Configuration

A DOTS client can negotiate, configure, and retrieve the DOTS signal channel session behavior with its DOTS peers. The DOTS signal channel can be used, for example, to configure the following:

a. Heartbeat interval (heartbeat-interval): DOTS agents regularly send heartbeats to each other after mutual authentication is successfully completed in order to keep the DOTS signal channel open. Heartbeat messages are exchanged between DOTS agents every 'heartbeat-interval' seconds to detect the current status of the DOTS signal channel session.

b. Missing heartbeats allowed (missing-hb-allowed): This variable
    indicates the maximum number of consecutive heartbeat messages
    for which a DOTS agent did not receive a response before
    concluding that the session is disconnected or defunct.

c. Acceptable signal loss ratio: Maximum retransmissions,
    retransmission timeout value, and other message transmission
    parameters for the DOTS signal channel.

When the DOTS signal channel is established over a reliable transport
(e.g., TCP), there is no need for the reliability mechanisms provided
by CoAP over UDP since the underlying TCP connection provides
retransmissions and deduplication [RFC8323]. As a reminder, CoAP
over reliable transports does not support Confirmable or Non-
confirmable message types. As such, the transmission-related
parameters (missing-hb-allowed and acceptable signal loss ratio) are
negotiated only for DOTS over unreliable transports.

The same or distinct configuration sets may be used during times when
a mitigation is active ('mitigating-config') and when no mitigation
is active ('idle-config'). This is particularly useful for DOTS
servers that might want to reduce heartbeat frequency or cease
heartbeat exchanges when an active DOTS client has not requested
mitigation. If distinct configurations are used, DOTS agents MUST
follow the appropriate configuration set as a function of the
mitigation activity (e.g., if no mitigation request is active (also
referred to as 'idle' time), 'idle-config'-related values must be
followed). Additionally, DOTS agents MUST automatically switch to
the other configuration upon a change in the mitigation activity
(e.g., if an attack mitigation is launched after an 'idle' time, the
DOTS agent switches from 'idle-config' to 'mitigating-config'-related
values).

The specification allows for a flexible retry configuration when an
unreliable transport is in use. For example, a server may be tweaked
to return the following configuration to be used when a mitigation is
active:

o a 'max-retransmit' set to '1' together with a higher 'missing-hb-
   allowed' value (e.g., 34) and a default 'ack-timeout' set to 2
   seconds. This configuration implies more frequent heartbeats in a
   given time span when a loss is encountered.

o a lower 'missing-hb-allowed' (e.g., 7) value but delegate the
   retransmission (with exponential back-off) to the underlying CoAP
   library by setting 'max-retransmit' to a high value (e.g., 3).
   When a loss is encountered, this configuration implies less
   frequent heartbeats compared to the previous bullet.

o a higher 'ack-timeout' value (e.g., 10 seconds), a 'max-
   retransmit' set to '1', and a 'missing-hb-allowed' value set to 7.
   Compared to the previous bullet, this configuration reduces by 50%
   the number of required heartbeats from the first transmission of a
   heartbeat message to the time when the DOTS agent gives up.

o etc.

CoAP Requests and responses are indicated for reliable delivery by
marking them as Confirmable messages. DOTS signal channel session
configuration requests and responses are marked as Confirmable
messages. As explained in Section 2.1 of [RFC7252], a Confirmable
message is retransmitted using a default timeout and exponential
back-off between retransmissions, until the DOTS server sends an
Acknowledgement message (ACK) with the same Message ID conveyed from
the DOTS client.

Message transmission parameters are defined in Section 4.8 of
[RFC7252]. The DOTS server can either piggyback the response in the
acknowledgement message or, if the DOTS server cannot respond
immediately to a request carried in a Confirmable message, it simply
responds with an Empty Acknowledgement message so that the DOTS
client can stop retransmitting the request. Empty Acknowledgement
messages are explained in Section 2.2 of [RFC7252]. When the
response is ready, the server sends it in a new Confirmable message
which in turn needs to be acknowledged by the DOTS client (see
Sections 5.2.1 and 5.2.2 of [RFC7252]). Requests and responses
exchanged between DOTS agents during 'idle' time are marked as
Confirmable messages.

   Implementation Note: A DOTS client that receives a response in a
   Confirmable message may want to clean up the message state right
   after sending the ACK. If that ACK is lost and the DOTS server
   retransmits the Confirmable message, the DOTS client may no longer
   have any state that would help it correlate this response: from
   the DOTS client's standpoint, the retransmission message is
   unexpected. The DOTS client will send a Reset message so it does
   not receive any more retransmissions. This behavior is normal and
   not an indication of an error (see Section 5.3.2 of [RFC7252] for
   more details).

4.5.1. Discover Configuration Parameters

A GET request is used to obtain acceptable (e.g., minimum and maximum
values) and current configuration parameters on the DOTS server for
DOTS signal channel session configuration. This procedure occurs
between a DOTS client and its immediate peer DOTS server. As such,
this GET request MUST NOT be relayed by a DOTS gateway.

Figure 18 shows how to obtain acceptable configuration parameters for
the DOTS server.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "config"
```

Figure 18: GET to Retrieve Configuration

The DOTS server in the 2.05 (Content) response conveys the current,
minimum, and maximum attribute values acceptable by the DOTS server
(Figure 19).

```
{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "missing-hb-allowed": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "max-retransmit": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "ack-timeout": {
        "max-value-decimal": "string",
        "min-value-decimal": "string",
        "current-value-decimal": "string"
      },
      "ack-random-factor": {
        "max-value-decimal": "string",
        "min-value-decimal": "string",
        "current-value-decimal": "string"
      }
    },
    "idle-config": {
      "heartbeat-interval": {
        "max-value": number,
        "min-value": number,
        "current-value": number
```

```
        },
        "missing-hb-allowed": {
          "max-value": number,
          "min-value": number,
          "current-value": number
        },
        "max-retransmit": {
          "max-value": number,
          "min-value": number,
          "current-value": number
        },
        "ack-timeout": {
          "max-value-decimal": "string",
          "min-value-decimal": "string",
          "current-value-decimal": "string"
        },
        "ack-random-factor": {
          "max-value-decimal": "string",
          "min-value-decimal": "string",
          "current-value-decimal": "string"
        }
      }
    }
  }
```

           Figure 19: GET Configuration Response Body Schema

   The parameters in Figure 19 are described below:

   mitigating-config: Set of configuration parameters to use when a
      mitigation is active. The following parameters may be included:

      heartbeat-interval: Time interval in seconds between two
         consecutive heartbeat messages.

         '0' is used to disable the heartbeat mechanism.

         This is an optional attribute.

      missing-hb-allowed: Maximum number of consecutive heartbeat
         messages for which the DOTS agent did not receive a response
         before concluding that the session is disconnected.

         This is an optional attribute.

      max-retransmit: Maximum number of retransmissions for a message
         (referred to as MAX_RETRANSMIT parameter in CoAP).

This is an optional attribute.

ack-timeout: Timeout value in seconds used to calculate the
    initial retransmission timeout value (referred to as
    ACK_TIMEOUT parameter in CoAP).

    This is an optional attribute.

ack-random-factor: Random factor used to influence the timing of
    retransmissions (referred to as ACK_RANDOM_FACTOR parameter in
    CoAP).

    This is an optional attribute.

idle-config: Set of configuration parameters to use when no
    mitigation is active. This attribute has the same structure as
    'mitigating-config'.

Figure 20 shows an example of acceptable and current configuration
parameters on a DOTS server for DOTS signal channel session
configuration. The same acceptable configuration is used during
mitigation and idle times.

```
{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "max-value": 240,
        "min-value": 15,
        "current-value": 30
      },
      "missing-hb-allowed": {
        "max-value": 9,
        "min-value": 3,
        "current-value": 5
      },
      "max-retransmit": {
        "max-value": 15,
        "min-value": 2,
        "current-value": 3
      },
      "ack-timeout": {
        "max-value-decimal": "30.00",
        "min-value-decimal": "1.00",
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "max-value-decimal": "4.00",
```

```
            "min-value-decimal": "1.10",
            "current-value-decimal": "1.50"
          }
        },
        "idle-config": {
          "heartbeat-interval": {
            "max-value": 240,
            "min-value": 15,
            "current-value": 30
          },
          "missing-hb-allowed": {
            "max-value": 9,
            "min-value": 3,
            "current-value": 5
          },
          "max-retransmit": {
            "max-value": 15,
            "min-value": 2,
            "current-value": 3
          },
          "ack-timeout": {
            "max-value-decimal": "30.00",
            "min-value-decimal": "1.00",
            "current-value-decimal": "2.00"
          },
          "ack-random-factor": {
            "max-value-decimal": "4.00",
            "min-value-decimal": "1.10",
            "current-value-decimal": "1.50"
          }
        }
      }
    }
```

                Figure 20: Example of a Configuration Response Body

4.5.2. Convey DOTS Signal Channel Session Configuration

   A PUT request (Figures 21 and 22) is used to convey the configuration
   parameters for the signal channel (e.g., heartbeat interval, maximum
   retransmissions). Message transmission parameters for CoAP are
   defined in Section 4.8 of [RFC7252]. The RECOMMENDED values of
   transmission parameter values are ack-timeout (2 seconds), max-
   retransmit (3), ack-random-factor (1.5). In addition to those
   parameters, the RECOMMENDED specific DOTS transmission parameter
   values are 'heartbeat-interval' (30 seconds) and 'missing-hb-allowed'
   (5).

Note: heartbeat-interval should be tweaked to also assist DOTS
messages for NAT traversal (SIG-011 of [RFC8612]). According to
[RFC8085], keepalive messages must not be sent more frequently
than once every 15 seconds and should use longer intervals when
possible. Furthermore, [RFC4787] recommends NATs to use a state
timeout of 2 minutes or longer, but experience shows that sending
packets every 15 to 30 seconds is necessary to prevent the
majority of middleboxes from losing state for UDP flows. From
that standpoint, the RECOMMENDED minimum heartbeat-interval is 15
seconds and the RECOMMENDED maximum heartbeat-interval is 240
seconds. The recommended value of 30 seconds is selected to
anticipate the expiry of NAT state.

A heartbeat-interval of 30 seconds may be considered as too chatty
in some deployments. For such deployments, DOTS agents may
negotiate longer heartbeat-interval values to prevent any network
overload with too frequent keepalives.

Different heartbeat intervals can be defined for 'mitigating-
config' and 'idle-config' to reduce being too chatty during idle
times. If there is an on-path translator between the DOTS client
(standalone or part of a DOTS gateway) and the DOTS server, the
'mitigating-config' heartbeat-interval has to be smaller than the
translator session timeout. It is recommended that the 'idle-
config' heartbeat-interval is also smaller than the translator
session timeout to prevent translator traversal issues, or
disabled entirely. Means to discover the lifetime assigned by a
translator are out of scope.

Section 4.2 of [RFC7252] defines a "CoAP Ping" mechanism.
Concretely, the DOTS agent sends an Empty Confirmable message and the
peer DOTS agent will respond by sending a Reset message.

When a Confirmable "CoAP Ping" is sent, and if there is no response,
the "CoAP Ping" is retransmitted max-retransmit number of times by
the CoAP layer using an initial timeout set to a random duration
between ack-timeout and (ack-timeout*ack-random-factor) and
exponential back-off between retransmissions. By choosing the
recommended transmission parameters, the "CoAP Ping" will timeout
after 45 seconds. If the DOTS agent does not receive any response
from the peer DOTS agent for 'missing-hb-allowed' number of
consecutive "CoAP Ping" Confirmable messages, it concludes that the
DOTS signal channel session is disconnected. A DOTS client MUST NOT
transmit a "CoAP Ping" while waiting for the previous "CoAP Ping"
response from the same DOTS server.

If the DOTS agent wishes to change the default values of message
transmission parameters, it SHOULD follow the guidance given in

   Section 4.8.1 of [RFC7252]. The DOTS agents MUST use the negotiated
   values for message transmission parameters and default values for
   non-negotiated message transmission parameters.

   The signal channel session configuration is applicable to a single
   DOTS signal channel session between DOTS agents, so the 'cuid' Uri-
   Path MUST NOT be used.

```
  Header: PUT (Code=0.03)
  Uri-Path: ".well-known"
  Uri-Path: "dots"
  Uri-Path: "config"
  Uri-Path: "sid=123"
  Content-Format: "application/dots+cbor"

  {
   ...
  }
```

         Figure 21: PUT to Convey the DOTS Signal Channel Session
                           Configuration Data

   The additional Uri-Path parameter to those defined in Table 1 is as
   follows:

   sid: Session Identifier is an identifier for the DOTS signal channel
         session configuration data represented as an integer. This
         identifier MUST be generated by DOTS clients. 'sid' values MUST
         increase monotonically (when a new PUT is generated by a DOTS
         client to convey the configuration parameters for the signal
         channel).

         This is a mandatory attribute.

```
        {
          "ietf-dots-signal-channel:signal-config": {
            "mitigating-config": {
              "heartbeat-interval": {
                "current-value": number
              },
              "missing-hb-allowed": {
                "current-value": number
              },
              "max-retransmit": {
                "current-value": number
              },
              "ack-timeout": {
                "current-value-decimal": "string"
              },
              "ack-random-factor": {
                "current-value-decimal": "string"
              }
            },
            "idle-config": {
              "heartbeat-interval": {
                "current-value": number
              },
              "missing-hb-allowed": {
                "current-value": number
              },
              "max-retransmit": {
                "current-value": number
              },
              "ack-timeout": {
                "current-value-decimal": "string"
              },
              "ack-random-factor": {
                "current-value-decimal": "string"
              }
            }
          }
        }
```

         Figure 22: PUT to Convey the DOTS Signal Channel Session
                 Configuration Data (Message Body Schema)

   The meaning of the parameters in the CBOR body (Figure 22) is defined
   in Section 4.5.1.

   At least one of the attributes 'heartbeat-interval', 'missing-hb-
   allowed', 'max-retransmit', 'ack-timeout', and 'ack-random-factor'
   MUST be present in the PUT request. Note that 'heartbeat-interval',

'missing-hb-allowed', 'max-retransmit', 'ack-timeout', and 'ack-
random-factor', if present, do not need to be provided for both
'mitigating-config', and 'idle-config' in a PUT request.

The PUT request with a higher numeric 'sid' value overrides the DOTS
signal channel session configuration data installed by a PUT request
with a lower numeric 'sid' value. To avoid maintaining a long list
of 'sid' requests from a DOTS client, the lower numeric 'sid' MUST be
automatically deleted and no longer available at the DOTS server.

Figure 23 shows a PUT request example to convey the configuration
parameters for the DOTS signal channel. In this example, the
heartbeat mechanism is disabled when no mitigation is active, while
the heartbeat interval is set to '91' when a mitigation is active.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "config"
Uri-Path: "sid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "current-value": 91
      },
      "missing-hb-allowed": {
        "current-value": 3
      },
      "max-retransmit": {
        "current-value": 3
      },
      "ack-timeout": {
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "current-value-decimal": "1.50"
      }
    },
    "idle-config": {
      "heartbeat-interval": {
        "current-value": 0
      },
      "max-retransmit": {
        "current-value": 3
      },
      "ack-timeout": {
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "current-value-decimal": "1.50"
      }
    }
  }
}
```

Figure 23: PUT to Convey the Configuration Parameters

The DOTS server indicates the result of processing the PUT request
using CoAP response codes:

o If the request is missing a mandatory attribute, does not include
   a 'sid' Uri-Path, or contains one or more invalid or unknown
   parameters, 4.00 (Bad Request) MUST be returned in the response.

o If the DOTS server does not find the 'sid' parameter value
   conveyed in the PUT request in its configuration data and if the
   DOTS server has accepted the configuration parameters, then a
   response code 2.01 (Created) MUST be returned in the response.

o If the DOTS server finds the 'sid' parameter value conveyed in the
   PUT request in its configuration data and if the DOTS server has
   accepted the updated configuration parameters, 2.04 (Changed) MUST
   be returned in the response.

o If any of the 'heartbeat-interval', 'missing-hb-allowed', 'max-
   retransmit', 'target-protocol', 'ack-timeout', and 'ack-random-
   factor' attribute values are not acceptable to the DOTS server,
   4.22 (Unprocessable Entity) MUST be returned in the response.
   Upon receipt of this error code, the DOTS client SHOULD retrieve
   the maximum and minimum attribute values acceptable to the DOTS
   server (Section 4.5.1).

   The DOTS client may re-try and send the PUT request with updated
   attribute values acceptable to the DOTS server.

A DOTS client may issue a GET message with 'sid' Uri-Path parameter
to retrieve the negotiated configuration. The response does not need
to include 'sid' in its message body.

4.5.3. Configuration Freshness and Notifications

Max-Age Option (Section 5.10.5 of [RFC7252]) SHOULD be returned by a
DOTS server to associate a validity time with a configuration it
sends. This feature allows the update of the configuration data if a
change occurs at the DOTS server side. For example, the new
configuration may instruct a DOTS client to cease heartbeats or
reduce heartbeat frequency.

It is NOT RECOMMENDED to return a Max-Age Option set to 0.

Returning a Max-Age Option set to 2**32-1 is equivalent to
associating an infinite lifetime with the configuration.

If a non-zero value of Max-Age Option is received by a DOTS client,
it MUST issue a GET request with 'sid' Uri-Path parameter to retrieve
the current and acceptable configuration before the expiry of the
value enclosed in the Max-Age option. This request is considered by
the client and the server as a means to refresh the configuration

parameters for the signal channel. When a DDoS attack is active,
refresh requests MUST NOT be sent by DOTS clients and the DOTS server
MUST NOT terminate the (D)TLS session after the expiry of the value
returned in Max-Age Option.

If Max-Age Option is not returned in a response, the DOTS client
initiates GET requests to refresh the configuration parameters each
60 seconds (Section 5.10.5 of [RFC7252]). To prevent such overload,
it is RECOMMENDED that DOTS servers return a Max-Age Option in GET
responses. Considerations related to which value to use and how such
value is set, are implementation- and deployment-specific.

If an Observe Option set to 0 is included in the configuration
request, the DOTS server sends notifications of any configuration
change (Section 4.2 of [RFC7641]).

If a DOTS server detects that a misbehaving DOTS client does not
contact the DOTS server after the expiry of Max-Age and retrieve the
signal channel configuration data, it MAY terminate the (D)TLS
session. A (D)TLS session is terminated by the receipt of an
authenticated message that closes the connection (e.g., a fatal alert
(Section 6 of [RFC8446])).

4.5.4. Delete DOTS Signal Channel Session Configuration

A DELETE request is used to delete the installed DOTS signal channel
session configuration data (Figure 24).

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "config"
Uri-Path: "sid=123"
```

                    Figure 24: Delete Configuration

The DOTS server resets the DOTS signal channel session configuration
back to the default values and acknowledges a DOTS client's request
to remove the DOTS signal channel session configuration using 2.02
(Deleted) response code.

Upon bootstrapping or reboot, a DOTS client MAY send a DELETE request
to set the configuration parameters to default values. Such a
request does not include any 'sid'.

4.6. Redirected Signaling

   Redirected DOTS signaling is discussed in detail in Section 3.2.2 of
   [I-D.ietf-dots-architecture].

   If a DOTS server wants to redirect a DOTS client to an alternative
   DOTS server for a signal session, then the response code 5.03
   (Service Unavailable) will be returned in the response to the DOTS
   client.

   The DOTS server can return the error response code 5.03 in response
   to a request from the DOTS client or convey the error response code
   5.03 in a unidirectional notification response from the DOTS server.

   The DOTS server in the error response conveys the alternate DOTS
   server's FQDN, and the alternate DOTS server's IP address(es) values
   in the CBOR body (Figure 25).

```
{
  "ietf-dots-signal-channel:redirected-signal": {
    "alt-server": "string",
    "alt-server-record": [
       "string"
     ]
}
```

           Figure 25: Redirected Server Error Response Body Schema

   The parameters are described below:

   alt-server: FQDN of an alternate DOTS server.

      This is a mandatory attribute.

   alt-server-record: A list of IP addresses of an alternate DOTS
      server.

      This is an optional attribute.

   The DOTS server returns the Time to live (TTL) of the alternate DOTS
   server in a Max-Age Option. That is, the time interval that the
   alternate DOTS server may be cached for use by a DOTS client. A Max-
   Age Option set to 2**32-1 is equivalent to receiving an infinite TTL.
   This value means that the alternate DOTS server is to be used until
   the alternate DOTS server redirects the traffic with another 5.03
   response which encloses an alternate server.

A Max-Age Option set to '0' may be returned for redirecting
mitigation requests. Such value means that the redirection applies
only for the mitigation request in progress. Returning short TTL in
a Max-Age Option may adversely impact DOTS clients on slow links.
Returning short values should be avoided under such conditions.

If the alternate DOTS server TTL has expired, the DOTS client MUST
use the DOTS server(s), that was provisioned using means discussed in
Section 4.1. This fall back mechanism is triggered immediately upon
expiry of the TTL, except when a DDoS attack is active.

Requests issued by misbehaving DOTS clients which do not honor the
TTL conveyed in the Max-Age Option or react to explicit re-direct
messages can be rejected by DOTS servers.

Figure 26 shows a 5.03 response example to convey the DOTS alternate
server 'alt-server.example' together with its IP addresses
2001:db8:6401::1 and 2001:db8:6401::2.

```
{
  "ietf-dots-signal-channel:redirected-signal": {
    "alt-server": "alt-server.example",
    "alt-server-record": [
        "2001:db8:6401::1",
        "2001:db8:6401::2"
      ]
}
```

        Figure 26: Example of Redirected Server Error Response Body

When the DOTS client receives 5.03 response with an alternate server
included, it considers the current request as failed, but SHOULD try
re-sending the request to the alternate DOTS server. During a DDoS
attack, the DNS server may be the target of another DDoS attack,
alternate DOTS server's IP addresses conveyed in the 5.03 response
help the DOTS client skip DNS lookup of the alternate DOTS server, at
the cost of trusting the first DOTS server to provide accurate
information. The DOTS client can then try to establish a UDP or a
TCP session with the alternate DOTS server. The DOTS client MAY
implement a method to construct IPv4-embedded IPv6 addresses
[RFC6052]; this is required to handle the scenario where an IPv6-only
DOTS client communicates with an IPv4-only alternate DOTS server.

If the DOTS client has been redirected to a DOTS server to which it
has already communicated with within the last five (5) minutes, it
MUST ignore the redirection and try to contact other DOTS servers
listed in the local configuration or discovered using dynamic means
such as DHCP or SRV procedures [I-D.ietf-dots-server-discovery]. It

is RECOMMENDED that DOTS clients support means to alert
administrators about redirect loops.

4.7. Heartbeat Mechanism

To provide an indication of signal health and distinguish an 'idle'
signal channel from a 'disconnected' or 'defunct' session, the DOTS
agent sends a heartbeat over the signal channel to maintain its half
of the channel (also, aligned with the "consents" recommendation in
Section 6 of [RFC8085]). The DOTS agent similarly expects a
heartbeat from its peer DOTS agent, and may consider a session
terminated in the prolonged absence of a peer agent heartbeat.
Concretely, while the communication between the DOTS agents is
otherwise quiescent, the DOTS client will probe the DOTS server to
ensure it has maintained cryptographic state and vice versa. Such
probes can also keep firewalls and/or stateful translators bindings
alive. This probing reduces the frequency of establishing a new
handshake when a DOTS signal needs to be conveyed to the DOTS server.

DOTS servers MAY trigger their heartbeat requests immediately after
receiving heartbeat probes from peer DOTS clients. As a reminder, it
is the responsibility of DOTS clients to ensure that on-path
translators/firewalls are maintaining a binding so that the same
external IP address and/or port number is retained for the DOTS
signal channel session.

In case of a massive DDoS attack that saturates the incoming link(s)
to the DOTS client, all traffic from the DOTS server to the DOTS
client will likely be dropped, although the DOTS server receives
heartbeat requests in addition to DOTS messages sent by the DOTS
client. In this scenario, DOTS clients MUST behave differently to
handle message transmission and DOTS signal channel session
liveliness during link saturation:

   The DOTS client MUST NOT consider the DOTS signal channel session
   terminated even after a maximum 'missing-hb-allowed' threshold is
   reached. The DOTS client SHOULD keep on using the current DOTS
   signal channel session to send heartbeat requests over it, so that
   the DOTS server knows the DOTS client has not disconnected the
   DOTS signal channel session.

   After the maximum 'missing-hb-allowed' threshold is reached, the
   DOTS client SHOULD try to resume the (D)TLS session. The DOTS
   client SHOULD send mitigation requests over the current DOTS
   signal channel session, and in parallel, for example, try to
   resume the (D)TLS session or use 0-RTT mode in DTLS 1.3 to
   piggyback the mitigation request in the ClientHello message.

As soon as the link is no longer saturated, if traffic from the
DOTS server reaches the DOTS client over the current DOTS signal
channel session, the DOTS client can stop (D)TLS session
resumption or if (D)TLS session resumption is successful then
disconnect the current DOTS signal channel session.

If the DOTS server receives traffic from the peer DOTS client (e.g.,
peer DOTS client initiated heartbeats) but maximum 'missing-hb-
allowed' threshold is reached, the DOTS server MUST NOT consider the
DOTS signal channel session disconnected. The DOTS server MUST keep
on using the current DOTS signal channel session so that the DOTS
client can send mitigation requests over the current DOTS signal
channel session. In this case, the DOTS server can identify the DOTS
client is under attack and the inbound link to the DOTS client
(domain) is saturated. Furthermore, if the DOTS server does not
receive a mitigation request from the DOTS client, it implies the
DOTS client has not detected the attack or, if an attack mitigation
is in progress, it implies the applied DDoS mitigation actions are
not yet effective to handle the DDoS attack volume.

If the DOTS server does not receive any traffic from the peer DOTS
client during the time span required to exhaust the maximum 'missing-
hb-allowed' threshold, the DOTS server concludes the session is
disconnected. The DOTS server can then trigger pre-configured
mitigation requests for this DOTS client (if any).

In DOTS over UDP, heartbeat messages MUST be exchanged between the
DOTS agents using the "CoAP Ping" mechanism defined in Section 4.2 of
[RFC7252].

In DOTS over TCP, heartbeat messages MUST be exchanged between the
DOTS agents using the Ping and Pong messages specified in Section 5.4
of [RFC8323]. That is, the DOTS agent sends a Ping message and the
peer DOTS agent would respond by sending a single Pong message. The
sender of a Ping message has to allow up to 'heartbeat-interval'
seconds when waiting for a Pong reply. When a failure is detected by
a DOTS client, it proceeds with the session recovery following the
same approach as the one used for unreliable transports.

5. DOTS Signal Channel YANG Modules

This document defines a YANG [RFC7950] module for DOTS mitigation
scope, DOTS signal channel session configuration data, and DOTS
redirection signaling.

This YANG module (ietf-dots-signal-channel) defines the DOTS client
interaction with the DOTS server as seen by the DOTS client. A DOTS
server is allowed to update the non-configurable 'ro' entities in the

responses. This YANG module is not intended to be used via NETCONF/
RESTCONF for DOTS server management purposes; such module is out of
the scope of this document. It serves only to provide a data model
and encoding, but not a management data model.

A companion YANG module is defined to include a collection of types
defined by IANA: "iana-dots-signal-channel" (Section 5.2).

5.1. Tree Structure

This document defines the YANG module "ietf-dots-signal-channel"
(Section 5.3), which has the following tree structure. A DOTS signal
message can be a mitigation, a configuration, or a redirect message.

```
 module: ietf-dots-signal-channel
    +--rw dots-signal
       +--rw (message-type)?
          +--:(mitigation-scope)
          |  +--rw scope* [cuid mid]
          |     +--rw cdid?                 string
          |     +--rw cuid                  string
          |     +--rw mid                   uint32
          |     +--rw target-prefix*        inet:ip-prefix
          |     +--rw target-port-range*    [lower-port]
          |     |  +--rw lower-port    inet:port-number
          |     |  +--rw upper-port?   inet:port-number
          |     +--rw target-protocol*      uint8
          |     +--rw target-fqdn*          inet:domain-name
          |     +--rw target-uri*           inet:uri
          |     +--rw alias-name*           string
          |     +--rw lifetime?             int32
          |     +--rw trigger-mitigation?   boolean
          |     +--ro mitigation-start?     uint64
          |     +--ro status?               iana-signal:status
          |     +--ro conflict-information
          |     |  +--ro conflict-status?   iana-signal:conflict-status
          |     |  +--ro conflict-cause?    iana-signal:conflict-cause
          |     |  +--ro retry-timer?       uint32
          |     |  +--ro conflict-scope
          |     |     +--ro target-prefix*       inet:ip-prefix
          |     |     +--ro target-port-range*   [lower-port]
          |     |     |  +--ro lower-port       inet:port-number
          |     |     |  +--ro upper-port?      inet:port-number
          |     |     +--ro target-protocol*     uint8
          |     |     +--ro target-fqdn*         inet:domain-name
          |     |     +--ro target-uri*          inet:uri
          |     |     +--ro alias-name*          string
          |     |     +--ro acl-list* [acl-name]
```

```
   │   │         │ +--ro acl-name
   │   │         │ │  -> /ietf-data:dots-data/dots-client/acls/
   │   │         │ │     acl/name
   │   │         │ +--ro acl-type?
   │   │         │    -> /ietf-data:dots-data/dots-client/acls/
   │   │         │       acl/type
   │   │         +--ro mid?                  -> ../../../mid
   │   +--ro bytes-dropped?        yang:zero-based-counter64
   │   +--ro bps-dropped?          yang:gauge64
   │   +--ro pkts-dropped?         yang:zero-based-counter64
   │   +--ro pps-dropped?          yang:gauge64
   │   +--rw attack-status?        iana-signal:attack-status
   +--:(signal-config)
   │   +--rw sid                  uint32
   │   +--rw mitigating-config
   │   │   +--rw heartbeat-interval
   │   │   │  +--ro max-value?       uint16
   │   │   │  +--ro min-value?       uint16
   │   │   │  +--rw current-value?   uint16
   │   │   +--rw missing-hb-allowed
   │   │   │  +--ro max-value?       uint16
   │   │   │  +--ro min-value?       uint16
   │   │   │  +--rw current-value?   uint16
   │   │   +--rw max-retransmit
   │   │   │  +--ro max-value?       uint16
   │   │   │  +--ro min-value?       uint16
   │   │   │  +--rw current-value?   uint16
   │   │   +--rw ack-timeout
   │   │   │  +--ro max-value-decimal?       decimal64
   │   │   │  +--ro min-value-decimal?       decimal64
   │   │   │  +--rw current-value-decimal?   decimal64
   │   │   +--rw ack-random-factor
   │   │      +--ro max-value-decimal?       decimal64
   │   │      +--ro min-value-decimal?       decimal64
   │   │      +--rw current-value-decimal?   decimal64
   │   +--rw idle-config
   │      +--rw heartbeat-interval
   │      │  +--ro max-value?       uint16
   │      │  +--ro min-value?       uint16
   │      │  +--rw current-value?   uint16
   │      +--rw missing-hb-allowed
   │      │  +--ro max-value?       uint16
   │      │  +--ro min-value?       uint16
   │      │  +--rw current-value?   uint16
   │      +--rw max-retransmit
   │      │  +--ro max-value?       uint16
   │      │  +--ro min-value?       uint16
   │      │  +--rw current-value?   uint16
```

```
             │     +--rw ack-timeout
             │     │  +--ro max-value-decimal?       decimal64
             │     │  +--ro min-value-decimal?       decimal64
             │     │  +--rw current-value-decimal?   decimal64
             │     +--rw ack-random-factor
             │        +--ro max-value-decimal?       decimal64
             │        +--ro min-value-decimal?       decimal64
             │        +--rw current-value-decimal?   decimal64
             +--:(redirected-signal)
                +--ro alt-server                string
                +--ro alt-server-record*        inet:ip-address
```

5.2. IANA DOTS Signal Channel YANG Module

```
   <CODE BEGINS> file "iana-dots-signal-channel@2019-01-17.yang"
   module iana-dots-signal-channel {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-dots-signal-channel";
     prefix iana-signal;

     organization
       "IANA";
     contact
       "Internet Assigned Numbers Authority

        Postal: ICANN
              12025 Waterfront Drive, Suite 300
              Los Angeles, CA  90094-2536
              United States of America
        Tel:    +1 310 301 5800
        <mailto:iana@iana.org>";
     description
       "This module contains a collection of YANG data types defined
        by IANA and used for DOTS signal channel protocol.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";
```

```
      revision 2019-01-17 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: Distributed Denial-of-Service Open Threat
                     Signaling (DOTS) Signal Channel Specification";
      }

      typedef status {
        type enumeration {
          enum attack-mitigation-in-progress {
            value 1;
            description
              "Attack mitigation setup is in progress (e.g., changing
               the network path to re-route the inbound traffic
               to DOTS mitigator).";
          }
          enum attack-successfully-mitigated {
            value 2;
            description
              "Attack is being successfully mitigated (e.g., traffic
               is redirected to a DDoS mitigator and attack
               traffic is dropped or blackholed).";
          }
          enum attack-stopped {
            value 3;
            description
              "Attack has stopped and the DOTS client can
               withdraw the mitigation request.";
          }
          enum attack-exceeded-capability {
            value 4;
            description
              "Attack has exceeded the mitigation provider
               capability.";
          }
          enum dots-client-withdrawn-mitigation {
            value 5;
            description
              "DOTS client has withdrawn the mitigation
               request and the mitigation is active but
               terminating.";
          }
          enum attack-mitigation-terminated {
            value 6;
            description
              "Attack mitigation is now terminated.";
          }
```

```
         enum attack-mitigation-withdrawn {
           value 7;
           description
             "Attack mitigation is withdrawn.";
         }
         enum attack-mitigation-signal-loss {
           value 8;
           description
             "Attack mitigation will be triggered
              for the mitigation request only when
              the DOTS signal channel session is lost.";
         }
       }
       description
         "Enumeration for status reported by the DOTS server.";
     }

     typedef conflict-status {
       type enumeration {
         enum request-inactive-other-active {
           value 1;
           description
             "DOTS Server has detected conflicting mitigation
              requests from different DOTS clients.
              This mitigation request is currently inactive
              until the conflicts are resolved. Another
              mitigation request is active.";
         }
         enum request-active {
           value 2;
           description
             "DOTS Server has detected conflicting mitigation
              requests from different DOTS clients.
              This mitigation request is currently active.";
         }
         enum all-requests-inactive {
           value 3;
           description
             "DOTS Server has detected conflicting mitigation
              requests from different DOTS clients.  All
              conflicting mitigation requests are inactive.";
         }
       }
       description
         "Enumeration for conflict status.";
     }

     typedef conflict-cause {
```

```
      type enumeration {
        enum overlapping-targets {
          value 1;
          description
            "Overlapping targets. conflict-scope provides
             more details about the exact conflict.";
        }
        enum conflict-with-acceptlist {
          value 2;
          description
            "Conflicts with an existing accept-list.

             This code is returned when the DDoS mitigation
             detects that some of the source addresses/prefixes
             listed in the accept-list ACLs are actually
             attacking the target.";
        }
        enum cuid-collision {
          value 3;
          description
            "Conflicts with the cuid used by another
             DOTS client.";
        }
      }
      description
        "Enumeration for conflict causes.";
    }

    typedef attack-status {
      type enumeration {
        enum under-attack {
          value 1;
          description
            "The DOTS client determines that it is still under
             attack.";
        }
        enum attack-successfully-mitigated {
          value 2;
          description
            "The DOTS client determines that the attack is
             successfully mitigated.";
        }
      }
      description
        "Enumeration for attack status codes.";
    }
  }
  <CODE ENDS>
```

5.3. IETF DOTS Signal Channel YANG Module

   This module uses the common YANG types defined in [RFC6991] and types
   defined in [I-D.ietf-dots-data-channel].

```
   <CODE BEGINS> file "ietf-dots-signal-channel@2019-01-17.yang"
   module ietf-dots-signal-channel {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal-channel";
     prefix signal;

     import ietf-inet-types {
       prefix inet;
       reference "Section 4 of RFC 6991";
     }
     import ietf-yang-types {
       prefix yang;
       reference "Section 3 of RFC 6991";
     }
     import ietf-dots-data-channel {
       prefix ietf-data;
       reference
         "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling
                    (DOTS) Data Channel Specification";
     }
     import iana-dots-signal-channel {
       prefix iana-signal;
     }

     organization
       "IETF DDoS Open Threat Signaling (DOTS) Working Group";
     contact
       "WG Web:   <https://datatracker.ietf.org/wg/dots/>
        WG List:  <mailto:dots@ietf.org>

        Editor:  Konda, Tirumaleswar Reddy
                 <mailto:TirumaleswarReddy_Konda@McAfee.com>

        Editor:  Mohamed Boucadair
                 <mailto:mohamed.boucadair@orange.com>

        Author:  Prashanth Patil
                 <mailto:praspati@cisco.com>

        Author:  Andrew Mortensen
                 <mailto:amortensen@arbor.net>

        Author:  Nik Teague
```

```
                <mailto:nteague@verisign.com>";
    description
      "This module contains YANG definition for the signaling
       messages exchanged between a DOTS client and a DOTS server.

       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2019-01-17 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Distributed Denial-of-Service Open Threat
                   Signaling (DOTS) Signal Channel Specification";
    }

    /*
     * Groupings
     */

    grouping mitigation-scope {
      description
        "Specifies the scope of the mitigation request.";
      list scope {
        key "cuid mid";
        description
          "The scope of the request.";
        leaf cdid {
          type string;
          description
            "The cdid should be included by a server-domain
             DOTS gateway to propagate the client domain
             identification information from the
             gateway's client-facing-side to the gateway's
             server-facing-side, and from the gateway's
             server-facing-side to the DOTS server.

             It may be used by the final DOTS server
```

```
          for policy enforcement purposes.";
      }
      leaf cuid {
        type string;
        description
          "A unique identifier that is
          generated by a DOTS client to prevent
          request collisions.  It is expected that the
          cuid will remain consistent throughout the
          lifetime of the DOTS client.";
      }
      leaf mid {
        type uint32;
        description
          "Mitigation request identifier.

          This identifier must be unique for each mitigation
          request bound to the DOTS client.";
      }
      uses ietf-data:target;
      leaf-list alias-name {
        type string;
        description
          "An alias name that points to a resource.";
      }
      leaf lifetime {
        type int32;
        units "seconds";
        default "3600";
        description
          "Indicates the lifetime of the mitigation request.

          A lifetime of '0' in a mitigation request is an
          invalid value.

          A lifetime of negative one (-1) indicates indefinite
          lifetime for the mitigation request.";
      }
      leaf trigger-mitigation {
        type boolean;
        default "true";
        description
          "If set to 'false', DDoS mitigation will not be
          triggered unless the DOTS signal channel
          session is lost.";
      }
      leaf mitigation-start {
        type uint64;
```

```
            config false;
            description
              "Mitigation start time is represented in seconds
               relative to 1970-01-01T00:00:00Z in UTC time.";
          }
          leaf status {
            type iana-signal:status;
            config false;
            description
              "Indicates the status of a mitigation request.
               It must be included in responses only.";
          }
          container conflict-information {
            config false;
            description
              "Indicates that a conflict is detected.
               Must only be used for responses.";
            leaf conflict-status {
              type iana-signal:conflict-status;
              description
                "Indicates the conflict status.";
            }
            leaf conflict-cause {
              type iana-signal:conflict-cause;
              description
                "Indicates the cause of the conflict.";
            }
            leaf retry-timer {
              type uint32;
              units "seconds";
              description
                "The DOTS client must not re-send the
                 same request that has a conflict before the expiry of
                 this timer.";
            }
            container conflict-scope {
              description
                "Provides more information about the conflict scope.";
              uses ietf-data:target {
                when "../conflict-cause = 'overlapping-targets'";
              }
              leaf-list alias-name {
                when "../../conflict-cause = 'overlapping-targets'";
                type string;
                description
                  "Conflicting alias-name.";
              }
              list acl-list {
```

```
               when "../../conflict-cause = 'conflict-with-acceptlist'";
               key "acl-name";
               description
                 "List of conflicting ACLs as defined in the DOTS data
                  channel.  These ACLs are uniquely defined by
                  cuid and acl-name.";
               leaf acl-name {
                 type leafref {
                   path "/ietf-data:dots-data/ietf-data:dots-client/"
                       + "ietf-data:acls/ietf-data:acl/ietf-data:name";
                 }
                 description
                   "Reference to the conflicting ACL name bound to
                    a DOTS client.";
               }
               leaf acl-type {
                 type leafref {
                   path "/ietf-data:dots-data/ietf-data:dots-client/"
                       + "ietf-data:acls/ietf-data:acl/ietf-data:type";
                 }
                 description
                   "Reference to the conflicting ACL type bound to
                    a DOTS client.";
               }
             }
             leaf mid {
               when "../../conflict-cause = 'overlapping-targets'";
               type leafref {
                 path "../../../mid";
               }
               description
                 "Reference to the conflicting 'mid' bound to
                  the same DOTS client.";
             }
           }
         }
         leaf bytes-dropped {
           type yang:zero-based-counter64;
           units "bytes";
           config false;
           description
             "The total dropped byte count for the mitigation
              request since the attack mitigation is triggered.
              The count wraps around when it reaches the maximum value
              of counter64 for dropped bytes.";
         }
         leaf bps-dropped {
           type yang:gauge64;
```

```
              config false;
              description
                "The average number of dropped bits per second for
                 the mitigation request since the attack
                 mitigation is triggered. This should be over
                 five-minute intervals (that is, measuring bytes
                 into five-minute buckets and then averaging these
                 buckets over the time since the mitigation was
                 triggered).";
            }
            leaf pkts-dropped {
              type yang:zero-based-counter64;
              config false;
              description
                "The total number of dropped packet count for the
                 mitigation request since the attack mitigation is
                 triggered.  The count wraps around when it reaches
                 the maximum value of counter64 for dropped packets.";
            }
            leaf pps-dropped {
              type yang:gauge64;
              config false;
              description
                "The average number of dropped packets per second
                 for the mitigation request since the attack
                 mitigation is triggered. This should be over
                 five-minute intervals (that is, measuring packets
                 into five-minute buckets and then averaging these
                 buckets over the time since the mitigation was
                 triggered).";
            }
            leaf attack-status {
              type iana-signal:attack-status;
              description
                "Indicates the status of an attack as seen by the
                 DOTS client.";
            }
          }
        }
      }

      grouping config-parameters {
        description
          "Subset of DOTS signal channel session configuration.";
        container heartbeat-interval {
          description
            "DOTS agents regularly send heartbeats to each other
             after mutual authentication is successfully
             completed in order to keep the DOTS signal channel
```

```
       open.";
     leaf max-value {
       type uint16;
       units "seconds";
       config false;
       description
         "Maximum acceptable heartbeat-interval value.";
     }
     leaf min-value {
       type uint16;
       units "seconds";
       config false;
       description
         "Minimum acceptable heartbeat-interval value.";
     }
     leaf current-value {
       type uint16;
       units "seconds";
       default "30";
       description
         "Current heartbeat-interval value.

          '0' means that heartbeat mechanism is deactivated.";
     }
   }
   container missing-hb-allowed {
     description
       "Maximum number of missing heartbeats allowed.";
     leaf max-value {
       type uint16;
       config false;
       description
         "Maximum acceptable missing-hb-allowed value.";
     }
     leaf min-value {
       type uint16;
       config false;
       description
         "Minimum acceptable missing-hb-allowed value.";
     }
     leaf current-value {
       type uint16;
       default "5";
       description
         "Current missing-hb-allowed value.";
     }
   }
   container max-retransmit {
```

```
        description
          "Maximum number of retransmissions of a Confirmable
           message.";
        leaf max-value {
          type uint16;
          config false;
          description
            "Maximum acceptable max-retransmit value.";
        }
        leaf min-value {
          type uint16;
          config false;
          description
            "Minimum acceptable max-retransmit value.";
        }
        leaf current-value {
          type uint16;
          default "3";
          description
            "Current max-retransmit value.";
        }
      }
      container ack-timeout {
        description
          "Initial retransmission timeout value.";
        leaf max-value-decimal {
          type decimal64 {
            fraction-digits 2;
          }
          units "seconds";
          config false;
          description
            "Maximum ack-timeout value.";
        }
        leaf min-value-decimal {
          type decimal64 {
            fraction-digits 2;
          }
          units "seconds";
          config false;
          description
            "Minimum ack-timeout value.";
        }
        leaf current-value-decimal {
          type decimal64 {
            fraction-digits 2;
          }
          units "seconds";
```

```
            default "2";
            description
              "Current ack-timeout value.";
          }
        }
        container ack-random-factor {
          description
            "Random factor used to influence the timing of
             retransmissions.";
          leaf max-value-decimal {
            type decimal64 {
              fraction-digits 2;
            }
            config false;
            description
              "Maximum acceptable ack-random-factor value.";
          }
          leaf min-value-decimal {
            type decimal64 {
              fraction-digits 2;
            }
            config false;
            description
              "Minimum acceptable ack-random-factor value.";
          }
          leaf current-value-decimal {
            type decimal64 {
              fraction-digits 2;
            }
            default "1.5";
            description
              "Current ack-random-factor value.";
          }
        }
      }

    grouping signal-config {
      description
        "DOTS signal channel session configuration.";
      leaf sid {
        type uint32;
        mandatory true;
        description
          "An identifier for the DOTS signal channel
           session configuration data.";
      }
      container mitigating-config {
        description
```

```
            "Configuration parameters to use when a mitigation
             is active.";
          uses config-parameters;
        }
        container idle-config {
          description
            "Configuration parameters to use when no mitigation
             is active.";
          uses config-parameters;
        }
      }

      grouping redirected-signal {
        description
          "Grouping for the redirected signaling.";
        leaf alt-server {
          type string;
          config false;
          mandatory true;
          description
            "FQDN of an alternate server.";
        }
        leaf-list alt-server-record {
          type inet:ip-address;
          config false;
          description
            "List of records for the alternate server.";
        }
      }

      /*
       * Main Container for DOTS Signal Channel
       */

      container dots-signal {
        description
          "Main container for DOTS signal message.

           A DOTS signal message can be a mitigation, a configuration,
           or a redirected signal message.";
        choice message-type {
          description
            "Can be a mitigation, a configuration, or a redirect
             message.";
          case mitigation-scope {
            description
              "Mitigation scope of a mitigation message.";
            uses mitigation-scope;
```

```
        }
        case signal-config {
          description
            "Configuration message.";
          uses signal-config;
        }
        case redirected-signal {
          description
            "Redirected signaling.";
          uses redirected-signal;
        }
      }
    }
  }
  <CODE ENDS>
```

6. YANG/JSON Mapping Parameters to CBOR

    All parameters in the payload of the DOTS signal channel MUST be
    mapped to CBOR types as shown in Table 4 and are assigned an integer
    key to save space.

    o Note: Implementers must check that the mapping output provided by
       their YANG-to-CBOR encoding schemes is aligned with the content of
       Table 4. For example, some CBOR and JSON types for enumerations
       and the 64-bit quantities can differ depending on the encoder
       used.

    The CBOR key values are divided into two types: comprehension-
    required and comprehension-optional. DOTS agents can safely ignore
    comprehension-optional values they don't understand, but cannot
    successfully process a request if it contains comprehension-required
    values that are not understood. The 4.00 response SHOULD include a
    diagnostic payload describing the unknown comprehension-required CBOR
    key values. The initial set of CBOR key values defined in this
    specification are of type comprehension-required.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
ietf-dots-signal-channel:mitigation-scope	container	1	5 map	Object
scope	list	2	4 array	Array
cdid	string	3	3 text string	String
cuid	string	4	3 text string	String
mid	uint32	5	0 unsigned	Number

target-prefix	leaf-list	6	4 array	Array
	inet: ip-prefix		3 text string	String
target-port-range	list	7	4 array	Array
lower-port	inet: port-number	8	0 unsigned	Number
upper-port	inet: port-number	9	0 unsigned	Number
target-protocol	leaf-list	10	4 array	Array
	uint8		0 unsigned	Number
target-fqdn	leaf-list	11	4 array	Array
	inet: domain-name		3 text string	String
target-uri	leaf-list	12	4 array	Array
	inet:uri		3 text string	String
alias-name	leaf-list	13	4 array	Array
	string		3 text string	String
lifetime	int32	14	0 unsigned	Number
			1 negative	Number
mitigation-start	uint64	15	0 unsigned	String
status	enumeration	16	0 unsigned	String
conflict-information	container	17	5 map	Object
conflict-status	enumeration	18	0 unsigned	String
conflict-cause	enumeration	19	0 unsigned	String
retry-timer	uint32	20	0 unsigned	Number
conflict-scope	container	21	5 map	Object
acl-list	list	22	4 array	Array
acl-name	leafref	23	3 text string	String
acl-type	leafref	24	3 text string	String
bytes-dropped	yang:zero- based- counter64	25	0 unsigned	String
bps-dropped	yang:gauge64	26	0 unsigned	String
pkts-dropped	yang:zero- based- counter64	27	0 unsigned	String
pps-dropped	yang:gauge64	28	0 unsigned	String
attack-status	enumeration	29	0 unsigned	String
ietf-dots-signal- channel:signal-config	container	30	5 map	Object
sid	uint32	31	0 unsigned	Number
mitigating-config	container	32	5 map	Object
heartbeat-interval	container	33	5 map	Object
max-value	uint16	34	0 unsigned	Number
min-value	uint16	35	0 unsigned	Number
current-value	uint16	36	0 unsigned	Number
missing-hb-allowed	container	37	5 map	Object
max-retransmit	container	38	5 map	Object

ack-timeout	container	39	5 map	Object
ack-random-factor	container	40	5 map	Object
max-value-decimal	decimal64	41	6 tag 4 [-2, integer]	String
min-value-decimal	decimal64	42	6 tag 4 [-2, integer]	String
current-value-decimal	decimal64	43	6 tag 4 [-2, integer]	String
idle-config	container	44	5 map	Object
trigger-mitigation	boolean	45	7 bits 20	False
			7 bits 21	True
ietf-dots-signal-cha nnel:redirected-signal	container	46	5 map	Object
alt-server	string	47	3 text string	String
alt-server-record	leaf-list inet: ip-address	48	4 array 3 text string	Array String

Table 4: CBOR Key Values Used in DOTS Signal Channel Messages & Their
         Mappings to JSON and YANG

7. (D)TLS Protocol Profile and Performance Considerations

7.1. (D)TLS Protocol Profile

   This section defines the (D)TLS protocol profile of DOTS signal
   channel over (D)TLS and DOTS data channel over TLS.

   There are known attacks on (D)TLS, such as man-in-the-middle and
   protocol downgrade attacks. These are general attacks on (D)TLS and,
   as such, they are not specific to DOTS over (D)TLS; refer to the
   (D)TLS RFCs for discussion of these security issues. DOTS agents
   MUST adhere to the (D)TLS implementation recommendations and security
   considerations of [RFC7525] except with respect to (D)TLS version.
   Since DOTS signal channel encryption relying upon (D)TLS is virtually
   a green-field deployment, DOTS agents MUST implement only (D)TLS 1.2
   or later.

   When a DOTS client is configured with a domain name of the DOTS
   server, and connects to its configured DOTS server, the server may
   present it with a PKIX certificate. In order to ensure proper
   authentication, a DOTS client MUST verify the entire certification
   path per [RFC5280]. Additionally, the DOTS client MUST use [RFC6125]
   validation techniques to compare the domain name with the certificate
   provided. Certification authorities that issue DOTS server
   certificates SHOULD support the DNS-ID and SRV-ID identifier types.
   DOTS server SHOULD prefer the use of DNS-ID and SRV-ID over CN-ID

identifier types in certificate requests (as described in Section 2.3 of [RFC6125]) and the wildcard character '*' SHOULD NOT be included in the presented identifier. DOTS doesn't use URI-IDs for server identity verification.

A key challenge to deploying DOTS is the provisioning of DOTS clients, including the distribution of keying material to DOTS clients to enable the required mutual authentication of DOTS agents. Enrollment over Secure Transport (EST) [RFC7030] defines a method of certificate enrollment by which domains operating DOTS servers may provide DOTS clients with all the necessary cryptographic keying material, including a private key and a certificate to authenticate themselves. One deployment option is DOTS clients behave as EST clients for certificate enrollment from an EST server provisioned by the mitigation provider. This document does not specify which EST or other mechanism the DOTS client uses to achieve initial enrollment.

The Server Name Indication (SNI) extension [RFC6066] defines a mechanism for a client to tell a (D)TLS server the name of the server it wants to contact. This is a useful extension for hosting environments where multiple virtual servers are reachable over a single IP address. The DOTS client may or may not know if it is interacting with a DOTS server in a virtual server hosting environment, so the DOTS client SHOULD include the DOTS server FQDN in the SNI extension.

Implementations compliant with this profile MUST implement all of the following items:

o DTLS record replay detection (Section 3.3 of [RFC6347]) or an equivalent mechanism to protect against replay attacks.

o DTLS session resumption without server-side state to resume session and convey the DOTS signal.

o At least one of raw public keys [RFC7250] or PSK handshake [RFC4279] with (EC)DHE key exchange which reduces the size of the ServerHello, and can be used by DOTS agents that cannot obtain certificates.

Implementations compliant with this profile SHOULD implement all of the following items to reduce the delay required to deliver a DOTS signal channel message:

o TLS False Start [RFC7918] which reduces round-trips by allowing the TLS client's second flight of messages (ChangeCipherSpec) to also contain the DOTS signal. TLS False Start is formally defined

for use with TLS, but the same technique is applicable to DTLS as well.

o Cached Information Extension [RFC7924] which avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake.

Compared to UDP, DOTS signal channel over TCP requires an additional round-trip time (RTT) of latency to establish a TCP connection. DOTS implementations are encouraged to implement TCP Fast Open [RFC7413] to eliminate that RTT.

7.2. (D)TLS 1.3 Considerations

TLS 1.3 provides critical latency improvements for connection establishment over TLS 1.2. The DTLS 1.3 protocol [I-D.ietf-tls-dtls13] is based upon the TLS 1.3 protocol and provides equivalent security guarantees. (D)TLS 1.3 provides two basic handshake modes the DOTS signal channel can take advantage of:

o A full handshake mode in which a DOTS client can send a DOTS mitigation request message after one round trip and the DOTS server immediately responds with a DOTS mitigation response. This assumes no packet loss is experienced.

o 0-RTT mode in which the DOTS client can authenticate itself and send DOTS mitigation request messages in the first message, thus reducing handshake latency. 0-RTT only works if the DOTS client has previously communicated with that DOTS server, which is very likely with the DOTS signal channel.

   The DOTS client has to establish a (D)TLS session with the DOTS server during 'idle' time and share a PSK.

   During a DDoS attack, the DOTS client can use the (D)TLS session to convey the DOTS mitigation request message and, if there is no response from the server after multiple retries, the DOTS client can resume the (D)TLS session in 0-RTT mode using PSK.

   DOTS servers that support (D)TLS 1.3 MAY allow DOTS clients to send early data (0-RTT). DOTS clients MUST NOT send "CoAP Ping" as early data; such messages MUST be rejected by DOTS servers. Section 8 of [RFC8446] discusses some mechanisms to implement to limit the impact of replay attacks on 0-RTT data. If the DOTS server accepts 0-RTT, it MUST implement one of these mechanisms to prevent replay at the TLS layer. A DOTS server can reject 0-RTT by sending a TLS HelloRetryRequest.

The DOTS signal channel messages sent as early data by the DOTS
client are idempotent requests. As a reminder, the Message ID
(Section 3 of [RFC7252]) is changed each time a new CoAP request
is sent, and the Token (Section 5.3.1 of [RFC7252]) is randomized
in each CoAP request. The DOTS server(s) MUST use the Message ID
and the Token in the DOTS signal channel message to detect replay
of early data at the application layer, and accept 0-RTT data at
most once from the same DOTS client. This anti-replay defense
requires sharing the Message ID and the Token in the 0-RTT data
between DOTS servers in the DOTS server domain. DOTS servers do
not rely on transport coordinates to identify DOTS peers. As
specified in Section 4.4.1, DOTS servers couple the DOTS signal
channel sessions using the DOTS client identity and optionally the
'cdid' parameter value. Furthermore, 'mid' value is monotonically
increased by the DOTS client for each mitigation request,
attackers replaying mitigation requests with lower numeric 'mid'
values and overlapping scopes with mitigation requests having
higher numeric 'mid' values will be rejected systematically by the
DOTS server. Likewise, 'sid' value is monotonically increased by
the DOTS client for each configuration request (Section 4.5.2),
attackers replaying configuration requests with lower numeric
'sid' values will be rejected by the DOTS server if it maintains a
higher numeric 'sid' value for this DOTS client.

Owing to the aforementioned protections, all DOTS signal channel
requests are safe to transmit in TLS 1.3 as early data. Refer to
[I-D.boucadair-dots-earlydata] for more details.

A simplified TLS 1.3 handshake with 0-RTT DOTS mitigation request
message exchange is shown in Figure 27.

```
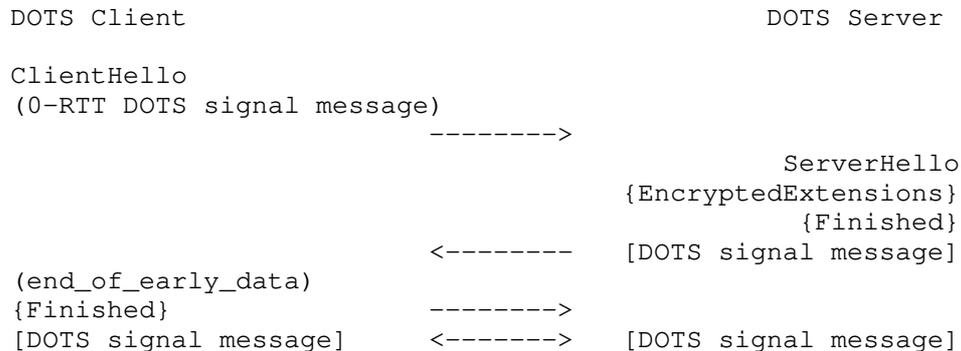        DOTS Client                                    DOTS Server

        ClientHello
        (0-RTT DOTS signal message)
                                    -------->
                                                          ServerHello
                                                {EncryptedExtensions}
                                                           {Finished}
                                    <--------   [DOTS signal message]
        (end_of_early_data)
        {Finished}                  -------->
        [DOTS signal message]       <------->   [DOTS signal message]
```

   Note that:
        () Indicates messages protected 0-RTT keys
        {} Indicates messages protected using handshake keys
        [] Indicates messages protected using 1-RTT keys

        Figure 27: A Simplified TLS 1.3 Handshake with 0-RTT

7.3. DTLS MTU and Fragmentation

   To avoid DOTS signal message fragmentation and the subsequent
   decreased probability of message delivery, DOTS agents MUST ensure
   that the DTLS record fit within a single datagram. As a reminder,
   DTLS handles fragmentation and reassembly only for handshake messages
   and not for the application data (Section 4.1.1 of [RFC6347]). If
   the PMTU cannot be discovered, DOTS agents MUST assume a PMTU of 1280
   bytes, as IPv6 requires that every link in the Internet have an MTU
   of 1280 octets or greater as specified in [RFC8200]. If IPv4 support
   on legacy or otherwise unusual networks is a consideration and the
   PMTU is unknown, DOTS implementations MAY assume on a PMTU of 576
   bytes for IPv4 datagrams, as every IPv4 host must be capable of
   receiving a packet whose length is equal to 576 bytes as discussed in
   [RFC0791] and [RFC1122].

   The DOTS client must consider the amount of record expansion expected
   by the DTLS processing when calculating the size of CoAP message that
   fits within the path MTU. Path MTU MUST be greater than or equal to
   [CoAP message size + DTLS 1.2 overhead of 13 octets + authentication
   overhead of the negotiated DTLS cipher suite + block padding]
   (Section 4.1.1.1 of [RFC6347]). If the total request size exceeds
   the path MTU then the DOTS client MUST split the DOTS signal into
   separate messages; for example, the list of addresses in the 'target-
   prefix' parameter could be split into multiple lists and each list
   conveyed in a new PUT request.

Implementation Note: DOTS choice of message size parameters works
well with IPv6 and with most of today's IPv4 paths. However, with
IPv4, it is harder to safely make sure that there is no IP
fragmentation. If the IPv4 path MTU is unknown, implementations may
want to limit themselves to more conservative IPv4 datagram sizes
such as 576 bytes, as per [RFC0791].

8. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

(D)TLS based upon client certificate can be used for mutual
authentication between DOTS agents. If, for example, a DOTS gateway
is involved, DOTS clients and DOTS gateways must perform mutual
authentication; only authorized DOTS clients are allowed to send DOTS
signals to a DOTS gateway. The DOTS gateway and the DOTS server must
perform mutual authentication; a DOTS server only allows DOTS signal
channel messages from an authorized DOTS gateway, thereby creating a
two-link chain of transitive authentication between the DOTS client
and the DOTS server.

The DOTS server should support certificate-based client
authentication. The DOTS client should respond to the DOTS server's
TLS CertificateRequest message with the PKIX certificate held by the
DOTS client. DOTS client certificate validation must be performed as
per [RFC5280] and the DOTS client certificate must conform to the
[RFC5280] certificate profile. If a DOTS client does not support TLS
client certificate authentication, it must support pre-shared key
based or raw public key based client authentication.

```
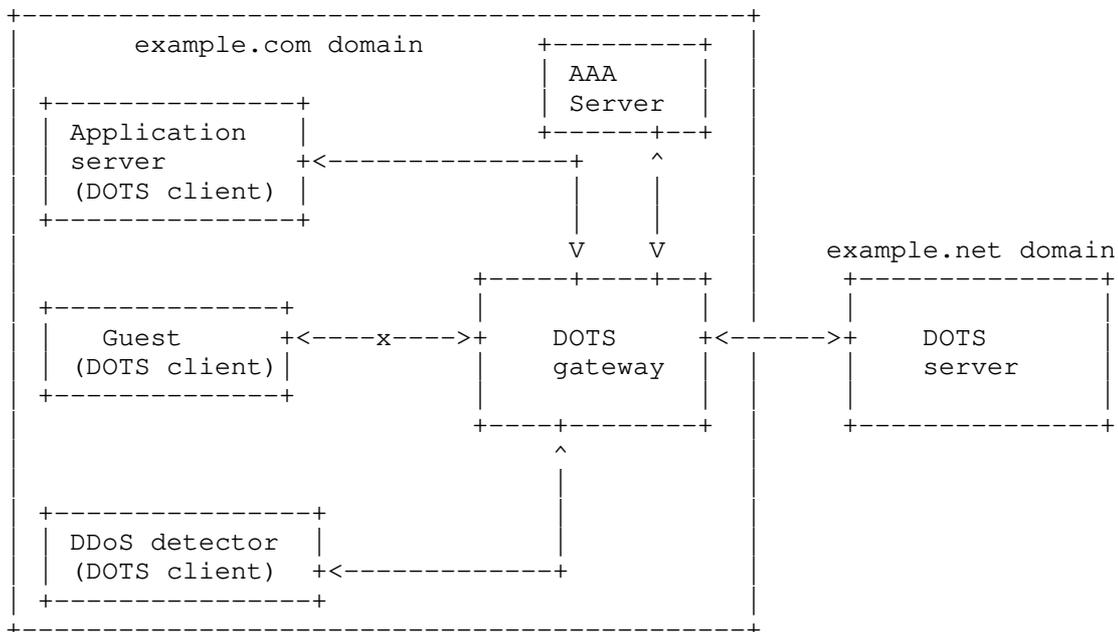+-----------------------------------------------+
|         example.com domain      +---------+   |
|                                 |  AAA    |   |
|                                 | Server  |   |
|  +---------------+              +------+--+   |
|  | Application   |              +<-----------+    ^       |
|  | server        +<--------------+      |         |
|  | (DOTS client) |              |       |         |
|  +---------------+              |       |         |
|                                 V       V    example.net domain
|                              +-----+----+--+      +---------------+
|  +-------------+             |             |      |               |
|  |   Guest     +<----x---->+ |    DOTS     +<------>+   DOTS      |
|  | (DOTS client)|           |   gateway   |      |   server      |
|  +-------------+             |             |      |               |
|                             +----+--------+ |     +---------------+
|                                  ^          |
|                                  |          |
|  +----------------+              |          |
|  | DDoS detector  |              |          |
|  | (DOTS client)  +<-------------+          |
|  +----------------+                         |
+-----------------------------------------------+
```

Figure 28: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 28, the DOTS gateway and DOTS
clients within the 'example.com' domain mutually authenticate. After
the DOTS gateway validates the identity of a DOTS client, it
communicates with the AAA server in the 'example.com' domain to
determine if the DOTS client is authorized to request DDoS
mitigation. If the DOTS client is not authorized, a 4.01
(Unauthorized) is returned in the response to the DOTS client. In
this example, the DOTS gateway only allows the application server and
DDoS attack detector to request DDoS mitigation, but does not permit
the user of type 'guest' to request DDoS mitigation.

Also, DOTS gateways and servers located in different domains must
perform mutual authentication (e.g., using certificates). A DOTS
server will only allow a DOTS gateway with a certificate for a
particular domain to request mitigation for that domain. In
reference to Figure 28, the DOTS server only allows the DOTS gateway
to request mitigation for 'example.com' domain and not for other
domains.

9. IANA Considerations

9.1. DOTS Signal Channel UDP and TCP Port Number

   IANA is requested to assign the port number TBD to the DOTS signal
   channel protocol for both UDP and TCP from the "Service Name and
   Transport Protocol Port Number Registry" available at
   https://www.iana.org/assignments/service-names-port-numbers/service-
   names-port-numbers.xhtml.

   The assignment of port number 4646 is strongly suggested, as 4646 is
   the ASCII decimal value for ".." (DOTS).

9.2. Well-Known 'dots' URI

   This document requests IANA to register the 'dots' well-known URI
   (Table 5) in the Well-Known URIs registry
   (https://www.iana.org/assignments/well-known-uris/well-known-
   uris.xhtml) as defined by [RFC5785]:

   +----------+----------------+--------------------+----------------+
   | URI | Change | Specification | Related |
   | suffix | controller | document(s) | information |
   +----------+----------------+--------------------+----------------+
   | dots | IETF | [RFCXXXX] | None |
   +----------+----------------+--------------------+----------------+

                      Table 5: 'dots' well-known URI

9.3. Media Type Registration

   This document requests IANA to register the "application/dots+cbor"
   media type in the "Media Types" registry [IANA.MediaTypes] in the
   manner described in [RFC6838], which can be used to indicate that the
   content is a DOTS signal channel object:

   o Type name: application
   o Subtype name: dots+cbor
   o Required parameters: N/A
   o Optional parameters: N/A
   o Encoding considerations: binary
   o Security considerations: See the Security Considerations section
      of [RFCXXXX]
   o Interoperability considerations: N/A
   o Published specification: [RFCXXXX]
   o Applications that use this media type: DOTS agents sending DOTS
      messages over CoAP over (D)TLS.
   o Fragment identifier considerations: N/A

o Additional information:

        Magic number(s): N/A
        File extension(s): N/A
        Macintosh file type code(s): N/A
o Person & email address to contact for further information:
   IESG, iesg@ietf.org
o Intended usage: COMMON
o Restrictions on usage: none
o Author: See Authors' Addresses section.
o Change controller: IESG
o Provisional registration? No

9.4. CoAP Content-Formats Registration

   This document requests IANA to register the CoAP Content-Format ID
   for the "application/dots+cbor" media type in the "CoAP Content-
   Formats" registry [IANA.CoAP.Content-Formats] (0-255 range):

o Media Type: application/dots+cbor
o Encoding: -
o Id: TBD1
o Reference: [RFCXXXX]

9.5. CBOR Tag Registration

   This section defines the DOTS CBOR tag as another means for
   applications to declare that a CBOR data structure is a DOTS signal
   channel object. Its use is optional and is intended for use in cases
   in which this information would not otherwise be known. DOTS CBOR
   tag is not required for DOTS signal channel protocol version
   specified in this document. If present, the DOTS tag MUST prefix a
   DOTS signal channel object.

   This document requests IANA to register the DOTS signal channel CBOR
   tag in the "CBOR Tags" registry [IANA.CBOR.Tags] using the 24-255
   range:

o CBOR Tag: TBD2 (please assign the same value as the Content-
   Format)
o Data Item: DDoS Open Threat Signaling (DOTS) signal channel object
o Semantics: DDoS Open Threat Signaling (DOTS) signal channel
   object, as defined in [RFCXXXX]
o Description of Semantics: [RFCXXXX]

9.6. DOTS Signal Channel Protocol Registry

   The document requests IANA to create a new registry, entitled "DOTS
   Signal Channel Registry". The following sections define sub-
   registries.

9.6.1. DOTS Signal Channel CBOR Key Values Sub-Registry

   The document requests IANA to create a new sub-registry, entitled
   "DOTS Signal Channel CBOR Key Values".

   The structure of this sub-registry is provided in Section 9.6.1.1.
   Section 9.6.1.2 provides how the registry is initially populated with
   the values in Table 4.

9.6.1.1. Registration Template

   Parameter name:
      Parameter name as used in the DOTS signal channel.

   CBOR Key Value:
      Key value for the parameter. The key value MUST be an integer in
      the 1-65535 range. The key values of the comprehension-required
      range (0x0001 - 0x3FFF) and of the comprehension-optional range
      (0x8000 - 0xBFFF) are assigned by IETF Review (Section 4.8 of
      [RFC8126]). The key values of the comprehension-optional range
      (0x4000 - 0x7FFF) are assigned by Specification Required
      (Section 4.6 of [RFC8126]) and of the comprehension-optional range
      (0xC000 - 0xFFFF) are reserved for Private Use (Section 4.1 of
      [RFC8126]).

      Registration requests for the 0x4000 - 0x7FFF range are evaluated
      after a three-week review period on the dots-signal-reg-
      review@ietf.org mailing list, on the advice of one or more
      Designated Experts. However, to allow for the allocation of
      values prior to publication, the Designated Experts may approve
      registration once they are satisfied that such a specification
      will be published. New registration requests should be sent in
      the form of an email to the review mailing list; the request
      should use an appropriate subject (e.g., "Request to register CBOR
      Key Value for DOTS: example"). IANA will only accept new
      registrations from the Designated Experts, and will check that
      review was requested on the mailing list in accordance with these
      procedures.

      Within the review period, the Designated Experts will either
      approve or deny the registration request, communicating this
      decision to the review list and IANA. Denials should include an

explanation and, if applicable, suggestions as to how to make the
request successful. Registration requests that are undetermined
for a period longer than 21 days can be brought to the IESG's
attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes
determining whether the proposed registration duplicates existing
functionality, whether it is likely to be of general applicability
or whether it is useful only for a single use case, and whether
the registration description is clear. IANA must only accept
registry updates to the 0x4000 - 0x7FFF range from the Designated
Experts and should direct all requests for registration to the
review mailing list. It is suggested that multiple Designated
Experts be appointed. In cases where a registration decision
could be perceived as creating a conflict of interest for a
particular Expert, that Expert should defer to the judgment of the
other Experts.

CBOR Major Type:
   CBOR Major type and optional tag for the parameter.

Change Controller:
   For Standards Track RFCs, list the "IESG". For others, give the
   name of the responsible party. Other details (e.g., email
   address) may also be included.

Specification Document(s):
   Reference to the document or documents that specify the parameter,
   preferably including URIs that can be used to retrieve copies of
   the documents. An indication of the relevant sections may also be
   included but is not required.

9.6.1.2. Initial Sub-Registry Content

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document(s)
ietf-dots-signal-channel:mitigation-scope	1	5	IESG	[RFCXXXX]
scope	2	4	IESG	[RFCXXXX]
cdid	3	3	IESG	[RFCXXXX]
cuid	4	3	IESG	[RFCXXXX]
mid	5	0	IESG	[RFCXXXX]
target-prefix	6	4	IESG	[RFCXXXX]
target-port-range	7	4	IESG	[RFCXXXX]
lower-port	8	0	IESG	[RFCXXXX]

upper-port	9	0	IESG	[RFCXXXX]
target-protocol	10	4	IESG	[RFCXXXX]
target-fqdn	11	4	IESG	[RFCXXXX]
target-uri	12	4	IESG	[RFCXXXX]
alias-name	13	4	IESG	[RFCXXXX]
lifetime	14	0/1	IESG	[RFCXXXX]
mitigation-start	15	0	IESG	[RFCXXXX]
status	16	0	IESG	[RFCXXXX]
conflict-information	17	5	IESG	[RFCXXXX]
conflict-status	18	0	IESG	[RFCXXXX]
conflict-cause	19	0	IESG	[RFCXXXX]
retry-timer	20	0	IESG	[RFCXXXX]
conflict-scope	21	5	IESG	[RFCXXXX]
acl-list	22	4	IESG	[RFCXXXX]
acl-name	23	3	IESG	[RFCXXXX]
acl-type	24	3	IESG	[RFCXXXX]
bytes-dropped	25	0	IESG	[RFCXXXX]
bps-dropped	26	0	IESG	[RFCXXXX]
pkts-dropped	27	0	IESG	[RFCXXXX]
pps-dropped	28	0	IESG	[RFCXXXX]
attack-status	29	0	IESG	[RFCXXXX]
ietf-dots-signal-channel:signal-config	30	5	IESG	[RFCXXXX]
sid	31	0	IESG	[RFCXXXX]
mitigating-config	32	5	IESG	[RFCXXXX]
heartbeat-interval	33	5	IESG	[RFCXXXX]
min-value	34	0	IESG	[RFCXXXX]
max-value	35	0	IESG	[RFCXXXX]
current-value	36	0	IESG	[RFCXXXX]
missing-hb-allowed	37	5	IESG	[RFCXXXX]
max-retransmit	38	5	IESG	[RFCXXXX]
ack-timeout	39	5	IESG	[RFCXXXX]
ack-random-factor	40	5	IESG	[RFCXXXX]
min-value-decimal	41	6tag4	IESG	[RFCXXXX]
max-value-decimal	42	6tag4	IESG	[RFCXXXX]
current-value-decimal	43	6tag4	IESG	[RFCXXXX]
idle-config	44	5	IESG	[RFCXXXX]
trigger-mitigation	45	7	IESG	[RFCXXXX]
ietf-dots-signal-channel:redirected-signal	46	5	IESG	[RFCXXXX]
alt-server	47	3	IESG	[RFCXXXX]
alt-server-record	48	4	IESG	[RFCXXXX]

Table 6: Initial DOTS Signal Channel CBOR Key Values Registry

9.6.2. Status Codes Sub-Registry

   The document requests IANA to create a new sub-registry, entitled
   "DOTS Signal Channel Status Codes". Codes in this registry are used
   as valid values of 'status' parameter.

   The registry is initially populated with the following values:

Code	Label	Description	Reference
1	attack-mitigation-in-progress	Attack mitigation setup is in progress (e.g., changing the network path to redirect the inbound traffic to a DOTS mitigator).	[RFCXXXX]
2	attack-successfully-mitigated	Attack is being successfully mitigated (e.g., traffic is redirected to a DDoS mitigator and attack traffic is dropped).	[RFCXXXX]
3	attack-stopped	Attack has stopped and the DOTS client can withdraw the mitigation request.	[RFCXXXX]
4	attack-exceeded-capability	Attack has exceeded the mitigation provider capability.	[RFCXXXX]

5	dots-client-withdrawn-mitigation	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.	[RFCXXXX]
6	attack-mitigation-terminated	Attack mitigation is now terminated.	[RFCXXXX]
7	attack-mitigation-withdrawn	Attack mitigation is withdrawn.	[RFCXXXX]
8	attack-mitigation-signal-loss	Attack mitigation will be triggered for the mitigation request only when the DOTS signal channel session is lost.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.6.3. Conflict Status Codes Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel Conflict Status Codes". Codes in this registry are used as valid values of 'conflict-status' parameter.

The registry is initially populated with the following values:

Code	Label	Description	Reference
1	request-inactive-other-active	DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently inactive until the conflicts are resolved. Another mitigation request is active.	[RFCXXXX]
2	request-active	DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently active.	[RFCXXXX]
3	all-requests-inactive	DOTS server has detected conflicting mitigation requests from different DOTS clients. All conflicting mitigation requests are inactive.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.6.4. Conflict Cause Codes Sub-Registry

   The document requests IANA to create a new sub-registry, entitled
   "DOTS Signal Channel Conflict Cause Codes". Codes in this registry
   are used as valid values of 'conflict-cause' parameter.

   The registry is initially populated with the following values:

   +------+------------------------+--------------------+-----------+
   | Code | Label | Description | Reference |
   +------+------------------------+--------------------+-----------+
1	overlapping-targets	Overlapping	[RFCXXXX]
		targets.	
2	conflict-with-acceptlist	Conflicts with an	[RFCXXXX]
		existing accept-	
		list. This code is	
		returned when the	
		DDoS mitigation	
		detects source	
		addresses/prefixes	
		in the accept-	
		listed ACLs are	
		attacking the	
		target.	
3	cuid-collision	CUID Collision.	[RFCXXXX]
		This code is	
		returned when a	
		DOTS client uses a	
		'cuid' that is	
		already used by	
		another DOTS	
		client.	
   +------+------------------------+--------------------+-----------+

   New codes can be assigned via Standards Action [RFC8126].

9.6.5. Attack Status Codes Sub-Registry

   The document requests IANA to create a new sub-registry, entitled
   "DOTS Signal Channel Attack Status Codes". Codes in this registry
   are used as valid values of 'attack-status' parameter.

   The registry is initially populated with the following values:

```
+------+----------------------------+---------------+-----------+
| Code | Label                      | Description   | Reference |
+------+----------------------------+---------------+-----------+
|  1   | under-attack               | The DOTS      | [RFCXXXX] |
|      |                            | client        |           |
|      |                            | determines    |           |
|      |                            | that it is    |           |
|      |                            | still under   |           |
|      |                            | attack.       |           |
|  2   | attack-successfully-mitigated | The DOTS   | [RFCXXXX] |
|      |                            | client        |           |
|      |                            | determines    |           |
|      |                            | that the      |           |
|      |                            | attack is     |           |
|      |                            | successfully  |           |
|      |                            | mitigated.    |           |
+------+----------------------------+---------------+-----------+
```

New codes can be assigned via Standards Action [RFC8126].

9.7. DOTS Signal Channel YANG Modules

This document requests IANA to register the following URIs in the
"ns" subregistry within the "IETF XML Registry" [RFC3688]:

    URI: urn:ietf:params:xml:ns:yang:ietf-dots-signal-channel
    Registrant Contact: The IESG.
    XML: N/A; the requested URI is an XML namespace.

    URI: urn:ietf:params:xml:ns:yang:iana-dots-signal-channel
    Registrant Contact: IANA.
    XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG modules in
the "YANG Module Names" subregistry [RFC7950] within the "YANG
Parameters" registry.

    Name: ietf-dots-signal-channel
    Namespace: urn:ietf:params:xml:ns:yang:ietf-dots-signal-channel
    Maintained by IANA: N
    Prefix: signal
    Reference: RFC XXXX

    Name: iana-dots-signal-channel
    Namespace: urn:ietf:params:xml:ns:yang:iana-dots-signal-channel
    Maintained by IANA: Y
    Prefix: iana-signal
    Reference: RFC XXXX

This document defines the initial version of the IANA-maintained
iana-dots-signal-channel YANG module. IANA is requested to add this
note:

   Status, conflict status, conflict cause, and attack status values
   must not be directly added to the iana-dots-signal-channel YANG
   module. They must instead be respectively added to the "DOTS
   Status Codes", "DOTS Conflict Status Codes", "DOTS Conflict Cause
   Codes", and "DOTS Attack Status Codes" registries.

When a 'status', 'conflict-status', 'conflict-cause', or 'attack-
status' value is respectively added to the "DOTS Status Codes", "DOTS
Conflict Status Codes", "DOTS Conflict Cause Codes", or "DOTS Attack
Status Codes" registry, a new "enum" statement must be added to the
iana-dots-signal-channel YANG module. The following "enum"
statement, and substatements thereof, should be defined:

"enum":   Replicates the label from the registry.

"value":   Contains the IANA-assigned value corresponding to the
                'status', 'conflict-status', 'conflict-cause', or
                'attack-status'.

"description": Replicates the description from the registry.

"reference":   Replicates the reference from the registry and add the
                title of the document.

When the iana-dots-signal-channel YANG module is updated, a new
"revision" statement must be added in front of the existing revision
statements.

IANA is requested to add this note to "DOTS Status Codes", "DOTS
Conflict Status Codes", "DOTS Conflict Cause Codes", and "DOTS Attack
Status Codes" registries:

   When this registry is modified, the YANG module iana-dots-signal-
   channel must be updated as defined in [RFCXXXX].

10. Security Considerations

High-level DOTS security considerations are documented in [RFC8612]
and [I-D.ietf-dots-architecture].

Authenticated encryption MUST be used for data confidentiality and
message integrity. The interaction between the DOTS agents requires
Datagram Transport Layer Security (DTLS) or Transport Layer Security
(TLS) with a cipher suite offering confidentiality protection, and

the guidance given in [RFC7525] MUST be followed to avoid attacks on
(D)TLS. The (D)TLS protocol profile used for the DOTS signal channel
is specified in Section 7.

If TCP is used between DOTS agents, an attacker may be able to inject
RST packets, bogus application segments, etc., regardless of whether
TLS authentication is used. Because the application data is TLS
protected, this will not result in the application receiving bogus
data, but it will constitute a DoS on the connection. This attack
can be countered by using TCP-AO [RFC5925]. Although not widely
adopted, if TCP-AO is used, then any bogus packets injected by an
attacker will be rejected by the TCP-AO integrity check and therefore
will never reach the TLS layer.

An attack vector that can be achieved if the 'cuid' is guessable is a
misbehaving DOTS client from within the client's domain which uses
the 'cuid' of another DOTS client of the domain to delete or alter
active mitigations. For this attack vector to happen, the
misbehaving client needs to pass the security validation checks by
the DOTS server, and eventually the checks of a client-domain DOTS
gateway.

A similar attack can be achieved by a compromised DOTS client which
can sniff the TLS 1.2 handshake, use the client certificate to
identify the 'cuid' used by another DOTS client. This attack is not
possible if algorithms such as version 4 Universally Unique
IDentifiers (UUIDs) in Section 4.4 of [RFC4122] are used to generate
the 'cuid' because such UUIDs are not a deterministic function of the
client certificate. Likewise, this attack is not possible with TLS
1.3 because most of the TLS handshake is encrypted and the client
certificate is not visible to eavesdroppers.

A compromised DOTS client can collude with a DDoS attacker to send
mitigation request for a target resource, gets the mitigation
efficacy from the DOTS server, and conveys the mitigation efficacy to
the DDoS attacker to possibly change the DDoS attack strategy.
Obviously, signaling an attack by the compromised DOTS client to the
DOTS server will trigger attack mitigation. This attack can be
prevented by monitoring and auditing DOTS clients to detect
misbehavior and to deter misuse, and by only authorizing the DOTS
client to request mitigation for specific target resources (e.g., an
application server is authorized to request mitigation for its IP
addresses but a DDoS mitigator can request mitigation for any target
resource in the network). Furthermore, DOTS clients are typically
co-located on network security services (e.g., firewall) and a
compromised security service potentially can do a lot more damage to
the network.

Rate-limiting DOTS requests, including those with new 'cuid' values, from the same DOTS client defends against DoS attacks that would result in varying the 'cuid' to exhaust DOTS server resources. Rate-limit policies SHOULD be enforced on DOTS gateways (if deployed) and DOTS servers.

In order to prevent leaking internal information outside a client-domain, DOTS gateways located in the client-domain SHOULD NOT reveal the identification information that pertains to internal DOTS clients (e.g., source IP address, client's hostname) unless explicitly configured to do so.

DOTS servers MUST verify that requesting DOTS clients are entitled to trigger actions on a given IP prefix. That is, only actions on IP resources that belong to the DOTS client' domain MUST be authorized by a DOTS server. The exact mechanism for the DOTS servers to validate that the target prefixes are within the scope of the DOTS client domain is deployment-specific.

The presence of DOTS gateways may lead to infinite forwarding loops, which is undesirable. To prevent and detect such loops, this document uses the Hop-Limit Option.

When FQDNs are used as targets, the DOTS server MUST rely upon DNS privacy enabling protocols (e.g., DNS over TLS [RFC7858] or DoH [RFC8484]) to prevent eavesdroppers from possibly identifying the target resources protected by the DDoS mitigation service, and means to ensure the target FQDN resolution is authentic (e.g., DNSSEC [RFC4034]).

CoAP-specific security considerations are discussed in Section 11 of [RFC7252], while CBOR-related security considerations are discussed in Section 8 of [RFC7049].

11. Contributors

The following individuals have contributed to this document:

o Jon Shallow, NCC Group, Email: jon.shallow@nccgroup.trust

o Mike Geller, Cisco Systems, Inc. 3250 Florida 33309 USA, Email: mgeller@cisco.com

o Robert Moskowitz, HTT Consulting Oak Park, MI 42837 United States, Email: rgm@htt-consult.com

o Dan Wing, Email: dwing-ietf@fuggles.com

12. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Roman Danyliw, Michael Richardson, Ehud Doron, Kaname Nishizuka, Dave Dolson, Liang Xia, Gilbert Clark, Xialiang Frank, Jim Schaad, Klaus Hartke, Nesredien Suleiman, Stephen Farrell, and Yoshifumi Nishida for the discussion and comments.

The authors would like to give special thanks to Kaname Nishizuka and Jon Shallow for their efforts in implementing the protocol and performing interop testing at IETF Hackathons.

Thanks to the core WG for the recommendations on Hop-Limit and redirect signaling.

Special thanks to Benjamin Kaduk for the detailed AD review.

Thanks to Alexey Melnikov, Adam Roach, Suresh Krishnan, Mirja Kuehlewind, and Alissa Cooper for the review.

13. References

13.1. Normative References

[I-D.ietf-core-hop-limit]
          Boucadair, M., K, R., and J. Shallow, "Constrained
          Application Protocol (CoAP) Hop-Limit Option", draft-ietf-
          core-hop-limit-04 (work in progress), July 2019.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
          DOI 10.17487/RFC0791, September 1981,
          <https://www.rfc-editor.org/info/rfc791>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
          Communication Layers", STD 3, RFC 1122,
          DOI 10.17487/RFC1122, October 1989,
          <https://www.rfc-editor.org/info/rfc1122>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
          DOI 10.17487/RFC3688, January 2004,
          <https://www.rfc-editor.org/info/rfc3688>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <https://www.rfc-editor.org/info/rfc3986>.

   [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key
              Ciphersuites for Transport Layer Security (TLS)",
              RFC 4279, DOI 10.17487/RFC4279, December 2005,
              <https://www.rfc-editor.org/info/rfc4279>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
              (CIDR): The Internet Address Assignment and Aggregation
              Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
              2006, <https://www.rfc-editor.org/info/rfc4632>.

   [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>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [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>.

   [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
              Uniform Resource Identifiers (URIs)", RFC 5785,
              DOI 10.17487/RFC5785, April 2010,
              <https://www.rfc-editor.org/info/rfc5785>.

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

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
              2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
              RFC 6991, DOI 10.17487/RFC6991, July 2013,
              <https://www.rfc-editor.org/info/rfc6991>.

   [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object
              Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
              October 2013, <https://www.rfc-editor.org/info/rfc7049>.

   [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
              Weiler, S., and T. Kivinen, "Using Raw Public Keys in
              Transport Layer Security (TLS) and Datagram Transport
              Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
              June 2014, <https://www.rfc-editor.org/info/rfc7250>.

   [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252,
              DOI 10.17487/RFC7252, June 2014,
              <https://www.rfc-editor.org/info/rfc7252>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <https://www.rfc-editor.org/info/rfc7641>.

   [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport
              Layer Security (TLS) False Start", RFC 7918,
              DOI 10.17487/RFC7918, August 2016,
              <https://www.rfc-editor.org/info/rfc7918>.

   [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security
              (TLS) Cached Information Extension", RFC 7924,
              DOI 10.17487/RFC7924, July 2016,
              <https://www.rfc-editor.org/info/rfc7924>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

   [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in
              the Constrained Application Protocol (CoAP)", RFC 7959,
              DOI 10.17487/RFC7959, August 2016,
              <https://www.rfc-editor.org/info/rfc7959>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

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

   [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", STD 86, RFC 8200,
              DOI 10.17487/RFC8200, July 2017,
              <https://www.rfc-editor.org/info/rfc8200>.

   [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2:
              Better Connectivity Using Concurrency", RFC 8305,
              DOI 10.17487/RFC8305, December 2017,
              <https://www.rfc-editor.org/info/rfc8305>.

   [RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K.,
              Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained
              Application Protocol) over TCP, TLS, and WebSockets",
              RFC 8323, DOI 10.17487/RFC8323, February 2018,
              <https://www.rfc-editor.org/info/rfc8323>.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

13.2. Informative References

   [I-D.boucadair-dots-earlydata]
              Boucadair, M. and R. K, "Using Early Data in DOTS", draft-
              boucadair-dots-earlydata-00 (work in progress), January
              2019.

[I-D.ietf-core-comi]
          Veillette, M., Stok, P., Pelov, A., Bierman, A., and I.
          Petrov, "CoAP Management Interface", draft-ietf-core-
          comi-07 (work in progress), July 2019.

[I-D.ietf-core-yang-cbor]
          Veillette, M., Petrov, I., and A. Pelov, "CBOR Encoding of
          Data Modeled with YANG", draft-ietf-core-yang-cbor-10
          (work in progress), April 2019.

[I-D.ietf-dots-architecture]
          Mortensen, A., K, R., Andreasen, F., Teague, N., and R.
          Compton, "Distributed-Denial-of-Service Open Threat
          Signaling (DOTS) Architecture", draft-ietf-dots-
          architecture-14 (work in progress), May 2019.

[I-D.ietf-dots-data-channel]
          Boucadair, M. and R. K, "Distributed Denial-of-Service
          Open Threat Signaling (DOTS) Data Channel Specification",
          draft-ietf-dots-data-channel-30 (work in progress), July
          2019.

[I-D.ietf-dots-multihoming]
          Boucadair, M., K, R., and W. Pan, "Multi-homing Deployment
          Considerations for Distributed-Denial-of-Service Open
          Threat Signaling (DOTS)", draft-ietf-dots-multihoming-02
          (work in progress), July 2019.

[I-D.ietf-dots-server-discovery]
          Boucadair, M. and R. K, "Distributed-Denial-of-Service
          Open Threat Signaling (DOTS) Server Discovery", draft-
          ietf-dots-server-discovery-04 (work in progress), June
          2019.

[I-D.ietf-dots-use-cases]
          Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
          Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
          Open Threat Signaling", draft-ietf-dots-use-cases-18 (work
          in progress), July 2019.

[I-D.ietf-tls-dtls13]
          Rescorla, E., Tschofenig, H., and N. Modadugu, "The
          Datagram Transport Layer Security (DTLS) Protocol Version
          1.3", draft-ietf-tls-dtls13-32 (work in progress), July
          2019.

[IANA.CBOR.Tags]
          IANA, "Concise Binary Object Representation (CBOR) Tags",
          <http://www.iana.org/assignments/cbor-tags/
          cbor-tags.xhtml>.

[IANA.CoAP.Content-Formats]
          IANA, "CoAP Content-Formats",
          <http://www.iana.org/assignments/core-parameters/
          core-parameters.xhtml#content-formats>.

[IANA.MediaTypes]
          IANA, "Media Types",
          <http://www.iana.org/assignments/media-types>.

[proto_numbers]
          "IANA, "Protocol Numbers"", 2011,
          <http://www.iana.org/assignments/protocol-numbers>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network
          Address Translator (Traditional NAT)", RFC 3022,
          DOI 10.17487/RFC3022, January 2001,
          <https://www.rfc-editor.org/info/rfc3022>.

[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>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
          Unique IDentifier (UUID) URN Namespace", RFC 4122,
          DOI 10.17487/RFC4122, July 2005,
          <https://www.rfc-editor.org/info/rfc4122>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
          Congestion Control Protocol (DCCP)", RFC 4340,
          DOI 10.17487/RFC4340, March 2006,
          <https://www.rfc-editor.org/info/rfc4340>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet
          Denial-of-Service Considerations", RFC 4732,
          DOI 10.17487/RFC4732, December 2006,
          <https://www.rfc-editor.org/info/rfc4732>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address
          Translation (NAT) Behavioral Requirements for Unicast
          UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
          2007, <https://www.rfc-editor.org/info/rfc4787>.

   [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol",
              RFC 4960, DOI 10.17487/RFC4960, September 2007,
              <https://www.rfc-editor.org/info/rfc4960>.

   [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common
              Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007,
              <https://www.rfc-editor.org/info/rfc4987>.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              DOI 10.17487/RFC5389, October 2008,
              <https://www.rfc-editor.org/info/rfc5389>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
              Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
              June 2010, <https://www.rfc-editor.org/info/rfc5925>.

   [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X.
              Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052,
              DOI 10.17487/RFC6052, October 2010,
              <https://www.rfc-editor.org/info/rfc6052>.

   [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful
              NAT64: Network Address and Protocol Translation from IPv6
              Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146,
              April 2011, <https://www.rfc-editor.org/info/rfc6146>.

   [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>.

   [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix
              Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011,
              <https://www.rfc-editor.org/info/rfc6296>.

   [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
              "Default Address Selection for Internet Protocol Version 6
              (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
              <https://www.rfc-editor.org/info/rfc6724>.

   [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
              Specifications and Registration Procedures", BCP 13,
              RFC 6838, DOI 10.17487/RFC6838, January 2013,
              <https://www.rfc-editor.org/info/rfc6838>.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
              DOI 10.17487/RFC6887, April 2013,
              <https://www.rfc-editor.org/info/rfc6887>.

   [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa,
              A., and H. Ashida, "Common Requirements for Carrier-Grade
              NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888,
              April 2013, <https://www.rfc-editor.org/info/rfc6888>.

   [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
              "Enrollment over Secure Transport", RFC 7030,
              DOI 10.17487/RFC7030, October 2013,
              <https://www.rfc-editor.org/info/rfc7030>.

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

   [RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson,
              "Architectural Considerations in Smart Object Networking",
              RFC 7452, DOI 10.17487/RFC7452, March 2015,
              <https://www.rfc-editor.org/info/rfc7452>.

   [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the
              NETCONF Protocol over Transport Layer Security (TLS) with
              Mutual X.509 Authentication", RFC 7589,
              DOI 10.17487/RFC7589, June 2015,
              <https://www.rfc-editor.org/info/rfc7589>.

   [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>.

   [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
              RFC 7951, DOI 10.17487/RFC7951, August 2016,
              <https://www.rfc-editor.org/info/rfc7951>.

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [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>.

   [RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
              Threat Signaling (DOTS) Requirements", RFC 8612,
              DOI 10.17487/RFC8612, May 2019,
              <https://www.rfc-editor.org/info/rfc8612>.

Appendix A. CUID Generation

   The document recommends the use of SPKI to generate the 'cuid'. This
   design choice is motivated by the following reasons:

   o SPKI is globally unique.

   o It is deterministic.

   o It allows to avoid extra cycles that may be induced by 'cuid'
      collision.

   o DOTS clients do not need to store the 'cuid' in a persistent
      storage.

   o It allows to detect compromised DOTS clients that do not adhere to
      the 'cuid' generation algorithm.

Authors' Addresses

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

   Email: kondtir@gmail.com

   Mohamed Boucadair (editor)
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI   48104
United States

Email: andrew@moretension.com

Nik Teague
Iron Mountain Data Centers
United Kingdom

Email: nteague@ironmountain.co.uk

DOTS   R. Dobbins
Internet-Draft   Arbor Networks
Intended status: Informational   D. Migault
Expires: April 28, 2018   Ericsson
                                                             S. Fouant

                                                          R. Moskowitz
                                                         HTT Consulting
                                                             N. Teague
                                                              Verisign
                                                                L. Xia
                                                                Huawei
                                                          K. Nishizuka
                                                     NTT Communications
                                                      October 25, 2017

                  Use cases for DDoS Open Threat Signaling
                        draft-ietf-dots-use-cases-08

Abstract

   The DDoS Open Threat Signaling (DOTS) effort is intended to provide a
   protocol that facilitates interoperability between multivendor
   solutions/services. This document presents use cases to evaluate the
   interactions expected between the DOTS components as well as DOTS
   messaging exchanges. The purpose of describing use cases is to
   identify the interacting DOTS components, how they collaborate and
   what are the types of information to be exchanged.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Currently, distributed denial-of-service (DDoS) attack mitigation
   solutions are largely based upon siloed, proprietary communications
   schemes which result in vendor lock-in. As a side-effect, this makes
   the configuration, provisioning, operation, and activation of these
   solutions a highly manual and often time-consuming process.
   Additionally, coordination of multiple DDoS mitigation solutions
   simultaneously engaged in defending the same organization (resources)
   against DDoS attacks is fraught with both technical and process-
   related hurdles. This greatly increase operational complexity and
   often results in suboptimal DDoS attack mitigation efficacy.

   The DDoS Open Threat Signaling (DOTS) effort is intended to specify a
   protocol that facilitates interoperability between multivendor DDoS
   mitigation solutions and ensures more automation in term of
   mitigation requests and attack characterization patterns. As DDoS
   solutions are broadly heterogeneous among different vendors, the
   primary goal for DOTS is to provide a high level interaction with
   these DDoS solutions such as initiating or terminating DDoS
   mitigation assistance.

   It should be noted that DOTS is not in and of itself intended to
   perform orchestration functions duplicative of the functionality
   being developed by the [I2NSF] WG; rather, DOTS is intended to allow
   devices, services, and applications to request DDoS attack mitigation
   assistance and receive mitigation status updates.

   These use cases are expected to provide inputs for the design of the
   DOTS protocol(s).

2. Terminology and Acronyms

   This document makes use of the terms defined in
   [I-D.ietf-dots-requirements].

   In addition, this document introduces the following terms:

Inter-domain: a DOTS communications relationship between distinct organizations with separate spans of administrative control. Typical inter-domain DOTS communication relationships would be between a DDoS mitigation service provider and an end-customer who requires DDoS mitigation assistance; between multiple DDoS mitigation service providers coordinating mutual defense of a mutual end-customer; or between DDoS mitigation service providers which are requesting additional DDoS mitigation assistance in for attacks which exceed their inherent DDoS mitigation capacities and/or capabilities.

Intra-domain: a DOTS communications relationship between various (network) elements that are owned and operated by the same administrative entity. A typical intra-domain DOTS communications relationship would be between DOTS agents [I-D.ietf-dots-requirements] within the same organization.

2.1. Requirements Terminology

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

3. Use Cases Scenarios

This section provides a high-level description of scenarios addressed by DOTS. In both sub-sections, the scenarios are provided in order to illustrate the use of DOTS in typical DDoS attack scenarios. They are not definitive, and other use cases are expected to emerge with widespread DOTS deployment.

All scenarios present a coordination between the targeted organization, the DDoS attack telemetry and the mitigator. The coordination and communication between these entities depends, for example, on the characteristic or functionality of the entity itself, the reliability of the information provided by DDoS attack telemetry, and the business relationship between the DDoS target domain and the mitigator.

More explicitly, in some cases, the DDoS attack telemetry may simply activate a DDoS mitigation, whereas in other cases, it may collaborate by providing some information about an attack. In some cases, the DDoS mitigation may be orchestrated, which includes selecting a specific appliance as well as starting/ending a mitigation.

3.1. Inter-domain Use Cases

   Inter-domain DOTS deployment scenarios span two or more distinct
   spans of administrative control. A typical inter-domain DOTS
   deployment may consist of an endpoint network such as an Internet-
   connected enterprise requesting DDoS mitigation assistance from one
   or more upstream transit providers offering DDoS mitigation services,
   or from a topologically-distant MSSP offering cloud-based overlay
   DDoS mitigation services. DOTS may also be used to facilitate
   coordination of DDoS mitigation activities between mitigation
   providers.

   Coordination between organizations making use of DOTS in such
   scenarios is necessary. Along with DOTS-specific tasks such as DOTS
   peering and validating the exchange of DOTS messaging between the
   relevant organizations, externalities relating to routing
   advertisements, authoritative DNS records (for DNS-based diversion),
   network access policies for DOTS nodes, service-level agreements
   (SLAs), and DDoS mitigation provisioning are required.

3.1.1. End-customer with a single upstream transit provider offering
        DDoS mitigation services

   In this scenario, an enterprise network with self-hosted Internet-
   facing properties such as Web servers, authoritative DNS servers, and
   VoIP PBXes has an intelligent DDoS mitigation system (IDMS) deployed
   to protect those servers and applications from DDoS attacks. In
   addition to their on-premise DDoS defense capability, they have
   contracted with their Internet transit provider for DDoS mitigation
   services which threaten to overwhelm their transit link bandwidth.

   The IDMS is configured such that if the incoming Internet traffic
   volume exceeds 50% of the provisioned upstream Internet transit link
   capacity, the IDMS will request DDoS mitigation assistance from the
   upstream transit provider.

   The requests to trigger, manage, and finalize a DDoS mitigation
   between the enterprise IDMS and the transit provider is performed
   using DOTS. The enterprise IDMS implements a DOTS client while the
   transit provider implements a DOTS server which is integrated with
   their DDoS mitigation orchestration system.

   When the IDMS detects an inbound DDoS attack targeting the enterprise
   servers and applications, it immediately begins mitigating the
   attack.

   During the course of the attack, the inbound traffic volume exceeds
   the 50% threshold; the IDMS DOTS client signals the DOTS server on

the upstream transit provider network to initiate DDoS mitigation.
The DOTS server signals the DOTS client that it can service this
request, and mitigation is initiated on the transit provider network.

Over the course of the attack, the DOTS server on the transit
provider network periodically signals the DOTS client on the
enterprise IDMS in order to provide mitigation status information,
statistics related to DDoS attack traffic mitigation, and related
information. Once the DDoS attack has ended, the DOTS server signals
the enterprise IDMS DOTS client that the attack has subsided.

The enterprise IDMS then requests that DDoS mitigation services on
the upstream transit provider network be terminated. The DOTS server
on the transit provider network receives this request, communicates
with the transit provider orchestration system controlling its DDoS
mitigation system to terminate attack mitigation, and once the
mitigation has ended, confirms the end of upstream DDoS mitigation
service to the enterprise IDMS DOTS client.

Note that communications between the enterprise DOTS client and the
upstream transit provider DOTS server may take place in-band within
the main Internet transit link between the enterprise and the transit
provider; out-of-band via a separate, dedicated wireline network link
utilized solely for DOTS signaling; or out-of-band via some other
form of network connectivity such as a third-party wireless 4G
network link.

(a) A DDoS attack is initiated against online properties of an
organization which has deployed DOTS-client-capable DDoS mitigators.

(b) CPE or PE DDoS mitigators detect, classify, and begin mitigating
the DDoS attack.

(c) CPE or PE DDoS mitigators determine that their capacity and/or
capability to mitigate the DDoS attack is insufficient, and utilize
their DOTS client functionality to send a DOTS mitigation service
initiation request to one or more DOTS servers residing on one or
more upstream transit networks, peer networks, or overlay MSSP
networks. This DOTS mitigation service initiation request may be
automatically initiated by the CPE or PE DDoS mitigators, or may be
manually triggered by personnel of the requesting organization in
response to an alert from the mitigators (the mechanism by which this
process takes place is beyond the scope of this document).

(d) The DOTS servers which receive the DOTS mitigation service
initiation requests determine that they have been configured to honor
requests from the requesting CPE or PE mitigators, and initiate
situationally-appropriate DDoS mitigation service on their respective

networks (the mechanism by which this process takes place is beyond the scope of this document).

(e) The DOTS servers transmit a DOTS service status message to the requesting CPE or PE mitigators indicating that upstream DDoS mitigation service has been initiated.

(f) While DDoS mitigation services are active, the DOTS servers regularly transmit DOTS mitigation status updates to the requesting CPE or PE mitigators.

(g) While DDoS mitigation services are active, the CPE or PE mitigators may optionally regularly transmit DOTS mitigation efficacy updates to the relevant DOTS servers.

(h) When the upstream DDoS mitigators determine that the DDoS attack has ceased, they indicate this change in status to their respective DOTS servers (the mechanism by which this process takes place is beyond the scope of this document).

(i) The DOTS servers transmit a DOTS mitigation status update to the CPE or PE mitigators indicating that the DDoS attack has ceased.

(j) The CPE or PE DDoS mitigators transmit a DOTS mitigation service termination request to the DOTS servers. This DOTS mitigation service termination request may be automatically initiated by the CPE or PE DDoS mitigators, or may be manually triggered by personnel of the requesting organization in response to an alert from the mitigators or a management system which monitors them (the mechanism by which this process takes place is beyond the scope of this document).

(k) The DOTS servers terminate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(l) The DOTS servers transmit a DOTS mitigation status update to the CPE or PE mitigators indicating that DDoS mitigation services have been terminated.

(m) The CPE or PE DDoS mitigators transmit a DOTS mitigation termination status acknowledgement to the DOTS servers.

3.1.2. End-customer with multiple upstream transit providers offering
        DDoS mitigation services

   This scenario shares many characteristics with the above, but with
   the key difference that the enterprise in question is multi-homed,
   i.e., has two or more upstream transit providers, and that they all
   provide DDoS mitigation services.

   In most cases, the communications model for a multi-homed model would
   be the same as in the single-homed model, merely duplicated in
   parallel. However, if two or more of the upstream transit providers
   have entered into a mutual DDoS mitigation agreement and have
   established DOTS peering between the participants, DDoS mitigation
   status messages may exchanged between the DOTS servers of the
   participants in order to provide a more complete picture of the DDoS
   attack scope, and allow for either automated or operator-assisted
   programmatic cooperative DDoS mitigation activities on the part of
   the transit providers.

   The DOTS communications between the upstream transit providers will
   consist of mitigation start, mitigation status, and mitigation
   termination messages.

3.1.3. End-customer with multiple upstream transit providers, but only
        a single upstream transit provider offering DDoS mitigation
        services

   This scenario is similar to the multi-homed scenario referenced
   above; however, only one of the upstream transit providers in
   question offers DDoS mitigation services. In this situation, the
   enterprise would cease advertising the relevant network prefixes via
   the transit providers which do not provide DDoS mitigation services -
   or, in the case where the enterprise does not control its own
   routing, request that the upstream transit providers which do not
   offer DDoS mitigation services stop advertising the relevant network
   prefixes on their behalf.

   Once it has been determined that the DDoS attack has ceased, the
   enterprise once again announces the relevant routes to the upstream
   transit providers which do not offer DDoS mitigation services, or
   requests that they resume announcing the relevant routes on behalf of
   the enterprise.

   Note that falling back to a single transit provider has the effect of
   reducing available inbound transit bandwidth during a DDoS attack.
   Without proper planning and sufficient provisioning of both the link
   capacity and DDoS mitigation capacity of the sole transit provider
   offering DDoS mitigation services, this reduction of available

bandwidth could lead to network link congestion caused by legitimate
inbound network traffic. Therefore, careful planning and
provisioning of both upstream transit bandwidth as well as DDoS
mitigation capacity is required in scenarios of this nature.

The withdrawal and announcement of routing prefixes described in this
use-case falls outside the scope of DOTS, although they could
conceivably be triggered as a result of provider-specific
orchestration triggered by the receipt of specific DOTS messages from
the enterprise in question.

The DOTS communications model for this scenario will be identical to
that described in Section 3.1.1.

3.1.4. End-customer with an overlay DDoS mitigation managed security
        service provider (MSSP)

This use case details an enterprise that has a local DDoS detection
and classification capability and may or may not have an on-premise
mitigation capability. The enterprise is contracted with an overlay
DDoS mitigation MSSP, topologically distant from the enterprise
network (i.e., not a direct upstream transit provider), which can
redirect (divert) traffic away from the enterprise, provide DDoS
mitigation services services, and then forward (re-inject) legitimate
traffic to the enterprise on an on-demand basis. In this scenario,
diversion of Internet traffic destined for the enterprise network
into the overlay DDoS mitigation MSSP network is typically
accomplished via eBGP announcements of the relevant enterprise
network CIDR blocks, or via authoritative DNS subdomain-based
mechanisms (other mechanisms are not precluded, these are merely the
most common ones in use today).

The enterprise determines thresholds at which a request for
mitigation is triggered indicating to the MSSP that inbound network
traffic should be diverted into the MSSP network and that DDoS
mitigation should be initiated. The enterprise may also elect to
manually request diversion and mitigation via the MSSP network as
desired.

The communications required to initiate, manage, and terminate active
DDoS mitigation by the MSSP is performed using DOTS. The enterprise
DDoS detection/classification system implements a DOTS client, while
the MSSP implements a DOTS server integrated with its DDoS mitigation
orchestration system. One or more out-of-band methods for initiating
a mitigation request, such as a Web portal, a smartphone app, or
voice support hotline, may also be made available by the MSSP.

   When an attack is detected, an automated or manual DOTS mitigation
   request is be generated by the enterprise and sent to the MSSP. The
   enterprise DOTS mitigation request is processed by the MSSP DOTS
   server, which validates the origin of the request and passes it to
   the MSSP DDoS mitigation orchestration system, which then initiates
   active DDoS mitigation. This action will usually involve the
   diversion of all network traffic destined for the targeted enterprise
   into the MSSP DDoS mitigation network, where it will be subjected to
   further scrutiny, with DDoS attack traffic filtered by the MSSP.
   Successful mitigation of the DDoS attack will not only result
   preserving the availability of services and applications resident on
   the enterprise network, but will also prevent DDoS attack traffic
   from ingressing the networks of the enterprise upstream transit
   providers/peers.

   The MSSP should signal via DOTS to the enterprise that a mitigation
   request has been received and acted upon, and should also include an
   update of the mitigation status. The MSSP may respond periodically
   with additional updates on the mitigation status to in order to
   enable the enterprise to make an informed decision on whether to
   maintain or terminate the mitigation. An alternative approach would
   be for the DOTS client mitigation request to include a time to live
   (TTL) for the mitigation, which may also be extended by the client
   should the attack still be ongoing as the TTL reaches expiration.

   A variation of this use case may be that the enterprise is providing
   a DDoS monitoring and analysis service to customers whose networks
   may be protected by any one of a number of third-party providers.
   The enterprise in question may integrate with these third-party
   providers using DOTS and signal accordingly when a customer is
   attacked - the MSSP may then manage the life-cycle of the attack
   mitigation on behalf of the enterprise.

   The DOTS communication model used in these scenarios will be
   identical to those described in Section 3.1.1 or Section 3.1.2.

3.1.5. End-customer operating an application or service with an
        integrated DOTS client

   In this scenario, a Web server has a built-in mechanism to detect and
   classify DDoS attacks, which also incorporates a DOTS client. When
   an attack is detected, the self-defense mechanism is activated, and
   local DDoS mitigation is initiated.

   The DOTS client built into the Web server has been configured to
   request DDoS mitigation services from an upstream transit provider or
   overlay MSSP once specific attack traffic thresholds have been
   reached, or certain network traffic conditions prevail. Once the

specified conditions have been met, the DOTS communications dialogue
and subsequent DDoS mitigation initiation and termination actions
described above take place.

The DOTS communication model used in these scenarios will be
identical to those described in Section 3.1.1, Section 3.1.2 or
Section 3.1.4, with the exception that the DOTS client will be the
application or service in question.

3.1.6. End-customer operating a CPE network infrastructure device with
        an integrated DOTS client

Similar to the above use-case featuring applications or services with
built-in DDoS attack detection/classification and DOTS client
capabilities, in this scenario, an end-customer network
infrastructure CPE device such as a router, layer-3 switch, firewall,
or load-balance incorporates both the functionality required to
detect and classify incoming DDoS attacks as well as DOTS client
functionality.

The subsequent DOTS communications dialogue and resultant DDoS
mitigation initiation and termination activities take place as
described in Section 3.1.1, Section 3.1.2 or Section 3.1.4.

3.1.7. End-customer with an out-of-band smartphone application
        featuring DOTS client capabilities

This scenario would typically apply in a small office/home office
(SOHO) setting, where the end-customer does not have CPE equipment or
software capable of detecting/classifying/mitigating DDoS attack, yet
still has a requirement for on-demand DDoS mitigation services. A
smartphone application containing a DOTS client would be provided by
the upstream transit mitigation provider or overlay DDoS MSSP to the
SOHO end-customer; this application would allow a manual 'panic-
button' to request the initiation and termination of DDoS mitigation
services.

The DOTS communications dialogue and resultant DDoS mitigation
initiation/status reporting/termination actions would then take place
as in as described in in Section 3.1.1, Section 3.1.2 or
Section 3.1.4, with the end-customer DOTS client application serving
to display received status information while DDoS mitigation
activities are taking place.

3.1.8. MSSP as an end-customer requesting overflow DDoS mitigation
        assistance from other MSSPs

   This is a more complex use-case involving multiple DDoS MSSPs,
   whether transit operators, overlay MSSPs, or both. In this scenario,
   an MSSP has entered into a pre-arranged DDoS mitigation assistance
   agreement with one or more other DDoS MSSPs in order to ensure that
   sufficient DDoS mitigation capacity and/or capabilities may be
   activated in the event that a given DDoS attack threatens to
   overwhelm the ability of a given DDoS MSSP to mitigate the attack on
   its own.

   BGP-based diversion (including relevant Letters of Authorization, or
   LoAs), DNS-based diversion (including relevant LoAs), traffic re-
   injection mechanisms such as Generic Routing Encapsulation (GRE)
   tunnels, provisioning of DDoS orchestration systems, et. al,. must be
   arranged in advance between the DDoS MSSPs which are parties to the
   agreement. They should also be tested on a regular basis.

   When a DDoS MSSP which is party to the agreement is nearing its
   capacity or ability to mitigate a given DDoS attack traffic, the DOTS
   client integrated with the MSSP DDoS mitigation orchestration system
   signals partner MSSPs to initiate network traffic diversion and DDoS
   mitigation activities. Ongoing attack and mitigation status messages
   may be passed between the DDoS MSSPs, and between the requesting MSSP
   and the ultimate end-customer of the attack.

   The DOTS dialogues and resultant DDoS mitigation-related activities
   in this scenario progress as described in the other use-cases
   detailed above. Once the requesting DDoS MSSP is confident that the
   DDoS attack has either ceased or has fallen to levels of traffic/
   complexity which they can handle on their own, the requesting DDoS
   MSSP DOTS client sends mitigation termination requests to the
   participating overflow DDoS MSSPs.

3.2. Intra-domain Use Cases

   While many of the DOTS-specific elements of inter-domain DOTS
   deployment scenarios apply to intra-domain scenarios, it is expected
   that many externalities such as coordination of and authorization for
   routing advertisements and authoritative DNS updates may be automated
   to a higher degree than is practicable in inter-domain scenarios,
   given that the scope of required activities and authorizations are
   confined to a single organization. In theory, provisioning and
   change-control related both to DOTS itself as well as relevant
   externalities may require less administrative overhead and less
   implementation lead-times.

The scope of potential DDoS mitigation actions may also be broader in intra-organizational scenarios, as presumably an organization will have a higher degree of autonomy with regards to both techically and administratively feasible activities.

3.2.1. Suppression of outbound DDoS traffic originating from a consumer broadband access network

While most DDoS defenses concentrate on inbound DDoS attacks ingressing from direct peering links or upstream transit providers, the DDoS attack traffic in question originates from one or more Internet-connected networks. In some cases, compromised devices residing on the local networks of broadband access customers are used to directly generate this DDoS attack traffic; in others, misconfigured devices residing on said local customer networks are exploited by attackers to launch reflection/amplification DDoS attacks. In either scenario, the outbound DDoS traffic emanating from these devices can be just as disruptive as an inbound DDoS attack, and can cause disruption for substantial proportions of the broadband access network operator's customer base.

Some broadband access network operators provide CPE devices (DSL modems/routers, cablemodems, FTTH routers, etc.) to their end-customers. Others allow end-customers to provide their own CPE devices. Many will either provide CPE devices or allow end-customers to supply their own.

Broadband access network operators typically have mechanisms to detect and classify both inbound and outbound DDoS attacks, utilizing flow telemetry exported from their peering/transit and customer aggregation routers. In the event of an outbound DDoS attack, they may make use of internally-developed systems which leverage their subscriber-management systems to de-provision end-customers who are sourcing outbound DDoS traffic; in some cases, they may have implemented quarantine systems to block all outbound traffic sourced from the offending end-customers. In either case, the perceived disruption of the end-customer's Internet access often prompts a help-desk call, which erodes the margins of the broadband access provider and can cause end-customer dissatisfaction.

Increasingly, CPE devices themselves are targeted by attackers who exploit security flaws in these devices in order to compromise them and subsume them into botnets, and then leverage them to launch outbound DDoS attacks. In all of the described scenarios, the end-customers are unaware that their computers and/or CPE devices have been compromised and are being used to launch outbound DDoS attacks - however, they may notice a degradation of their Internet connectivity as a result of outbound bandwidth consumption or other disruption.

By deploying DOTS-enabled telemetry systems and CPE devices (and possibly requiring DOTS functionality in customer-provided CPE devices), broadband access providers can utilize a standards-based mechanism to suppress outbound DDoS attack traffic while optionally allowing legitimate end-customer traffic to proceed unmolested.

In order to achieve this capability, the telemetry analysis system utilized by the broadband access provider must have DOTS client functionality, and the end-customer CPE devices must have DOTS server functionality. When the telemetry analysis system detects and classifies an outbound DDoS attack sourced from one or more end-customer networks/devices, the DOTS client of the telemetry analysis system sends a DOTS request to the DOTS server implemented on the CPE devices, requesting local mitigation assistance in suppressing either the identified outbound DDoS traffic, or all outbound traffic sourced from the end-customer networks/devices. The DOTS server residing within the CPE device(s) would then perform predefined actions such as implementing on-board access-control lists (ACLs) to suppress the outbound traffic in question and prevent it from leaving the local end-customer network(s).

Broadband access network operators may choose to implement a quarantine of all or selected network traffic originating from end-customer networks/devices which are sourcing outbound DDoS traffic, redirecting traffic from interactive applications such as Web browsers to an internal portal which informs the end-customer of the quarantine action, and providing instructions for self-remediation and/or helpdesk contact information.

Quarantine systems for broadband access networks are typically custom-developed and -maintained, and are generally deployed only by a relatively small number of broadband access providers with considerable internal software development and support capabilities. By requiring the manufacturers of operator-supplied CPE devices to implement DOTS server functionality, and requiring customer-provided CPE devices to feature DOTS server functionality, broadband access network operators who previously could not afford the development expense of creating custom quarantine systems to integrate DOTS-enabled network telemetry systems to act as DOTS clients and perform effective quarantine of end-customer networks and devices until such time as they have been remediated.

The DOTS communications model in this scenario resembles that described in Section 3.1.1, except that all the DOTS communications take place within the same span of administrative control and the same network.

3.2.2. Home Network DDoS Detection Collaboration for ISP network
        management

   Home networks run with (limited) bandwidth as well as limited routing
   resources, while they are expected to provide services reachable from
   the outside [RFC7368]. This makes such networks some easy targets to
   DDoS attacks via their WAN interface. As these DDoS attacks are easy
   to perform, they may remain undetected by the upstream ISP. When the
   CPE is congested, the customer is likely to call the ISP hotline. In
   order to improve the quality of experience of the connectivity as
   well as to automate the request for DDoS mitigation, ISPs are likely
   to consider a standard mean for CPEs to automatically inform a
   dedicated service mitigation platform when they are under a suspected
   DDoS.

   Note also that this section only considers DDoS attacks CPE or
   services in the home network are encountering. This differs from
   DDoS attacks the CPE or any device within the home network may take
   part of - such as botnets. In the later attacks, the home network
   generates traffic under the control of a botmaster. Such attacks may
   only be detected once the attacks have been characterized. It would
   be tempting to consider a feature in the DOTS protocol to allow a
   DOTS server to inform a CPE that some suspect traffic is being sent
   by the CPE so that appropriate actions are undertaken by the CPE/
   user. Nevertheless, this feature would require some interaction with
   the CPE administrator. Such scenario is outside the scope of this
   document.

   In this use case, ISPs are willing to prevent their customer
   undergoing DDoS attacks in order to enhance the quality of experience
   of their customers, to avoid unnecessary costly call on hot lines as
   well as to optimize the bandwidth of their network. A key challenge
   for the ISP is to detect DDoS attacks. In fact, DDoS detection is
   not only fine grained but is also expected to be different for each
   home network or small businesses networks (SOHO), and the ISP is
   unlikely to have sufficient resource to inspect the traffic of all
   its customers.

   In order to address these challenges, ISPs are delegating the DDoS
   detection to CPE of home network or SOHO. Outsourcing the detection
   on the CPE provides the following advantages to the ISP: 1) Avoid the
   ISP to dedicate a huge amount of resource for deep packet inspection
   over a large amount of traffic with a specific security policies
   associated to each home network. It is expected that such traffic
   only constitutes a small fraction of the total traffic the ISP is
   responsible for. 2) DDoS detection is deployed in a scalable way. 3)
   Provide more deterministic DDoS attack detection. For example, what
   could be suspected to be an UDP flood by the ISP may be consented by

the terminating point hosted in the home network or SOHO. In fact,
without specific home network security policies, the ISP is likely to
detect DDoS attack over regular traffic or to miss DDoS attacks
targeting a specific home network or CPE. In the first case, this
would result in the ISP spending unnecessary resources and in the
second case this would directly impact the quality of experience of
the customer.

Note that in this scenario slightly differs from the "Enterprise with
an upstream transit provider DDoS mitigation Service" scenario
described in Section 3.1.1. In this scenario, the detection DDoS is
motivated by the ISP in order to operate appropriately its network.

For that purpose, it requires some collaboration with the home
network. In Section 3.1.1, the target network requests a mitigation
service from the upstream transit provider in order to operate its
services.

Even though the motivations differ, there are still significant
advantages for the home network to collaborate. On the home network
or SOHO perspective such collaboration provides the following
advantages: 1) If it removes the flows contributing to a DDoS
attacks, then it enhances the quality of experience of the users of
the targeted services or the entire home network. 2) If mitigation is
being handled by the ISP rather then the home network, then it
reduces the management of DDoS attacks by the network administrator
which involves detection as well as mitigation as well as the
provisioning of extra resources. 3) If the DDoS detection is based on
information specific to the home network, such as for example the
description of the services, the hosts capacities or the network
topology, then performing the DDoS detection by the home network
instead of the ISP avoids the home network to leak private
information to the ISP. In that sense, it better preserves the home
network or SOHO privacy while enabling a better detection. However,
the request for mitigation may still leak some informations. ISPs
must not retrieve sensitive data without the consent of the user.
This is usually captured in administrative contracts that are out of
scope of this document.

When the CPE suspects an attack, it notifies automatically or the
ISP. The contact address of the DDoS Mitigation service of the ISP
may be hard coded or may be configured manually or automatically
(e.g., eventually the DHCP server may provide the DDoS mitigation
service via specific DHCP options).

The communication to trigger a DDoS mitigation between the home
network and the ISP is performed using DOTS. The home network CPE
implements a DOTS client while the ISP implements a DOTS server.

The DOTS client on the CPE monitors the status of CPE's resource and
WAN link bandwidth usage. If something unusual happens based on
preconfigured throughput, traffic patter, explicit action from the
user, or some heuristics methods, the DOTS client sends a DOTS
mitigation request to the ISP DOTS server. Typically, a default
configuration with no additional information associated to the DOTS
mitigation request is expected. The ISP derives traffic to mitigate
from the CPE IP address.

In some cases, the DOTS mitigation request contains options such as
some IP addresses or prefixes that belongs to a whitelist or a
blacklist. In this case, the white and black lists are not
associated to some analysis performed by the CPE - as the CPE is
clearly not expected to analyze such attacks. Instead these are part
of some configuration parameters. For example, in the case of small
business, one may indicate specific legitimate IP addresses such as
those used for VPNs, or third party services the company is likely to
set a session. Similarly, the CPE may provide the IP addresses
targeting the assets to be protected inside the network. Note that
the IP address is the IP address used to reach the asset from the
internet, and as such is expected to be globally routable. Such
options may include the IP address as well as a service description.
Similarly to the previous blacklist and whitelist, such information
are likely not derived from a traffic analysis performed by the CPE,
but instead are more related to configuration parameters.

Upon receiving the DOTS mitigation request, the DOTS server
acknowledges its reception and confirms DDoS mitigation starts or
not. Such feed back is mostly to avoid retransmission of the
request.

Note that the ISP is connected to multiple CPEs and as such the CPE
can potentially perform DDoS attack to the DOTS server. ISP may use
gateways to absorbs the traffic. These gateways, will typically
aggregate a smaller number of CPEs and retransmit to the destination
DOTS Server a selected information. Note that such gateways may
somehow act as a DOTS relay, which is implemented with a DOTS Server
and a DOTS Client. Note also that the case of a large DDoS attack
targeting simultaneously multiple CPEs is expected to be detected and
mitigated by the upstream architecture, eventually without DOTS
alerts sent by each single CPE.

ISP may activate mitigation for the traffic associated to the CPE
sending the alert or instead to the traffic associated to all CPE.
Such decisions are not part of DOTS, but instead depend on the
policies of the ISP.

It is unlikely the CPE will follow the status of the mitigation. The
ISP is only expected to inform the CPE the mitigation has been
stopped.

Upon receipt of such notification the CPE may, for example, re-
activate the monitoring jobs and thus is likely to provide some
further DOTS alert.

3.2.3. DDoS Orchestration

In this use case, one or multiple DDoS telemetry systems or
monitoring devices such as a flow telemetry collector monitor a
network -- typically an ISP network. Upon detection of a DDoS
attack, these telemetry systems alert an orchestrator in charge of
coordinating the various DDoS mitigation systems within the domain.
The telemetry systems may be configured to provide necessary and
useful pieces of information, such as a preliminary analysis of the
observation to the orchestrator.

The orchestrator analyses the various information it receives from
specialized equipement, and elaborates one or multiple DDoS
mitigation strategies. In some case, a manual confirmation may also
be required to choose a proposed strategy or to initiate a DDoS
mitigation. The DDoS mitigation may consist of multiple steps such
as configuring the network, various hardware, or updating already
instantiated DDoS mitigation functions. In some cases, some specific
virtual DDoS mitigation functions must be instantiated and properly
ordered. Eventually, the coordination of the mitigation may involve
external DDoS resources such as a transit provider (Section 3.1.1) or
an MSSP (Section 3.1.4).

The communications used to trigger a DDoS mitigation between the
telemetry and monitoring systems and the orchestrator is performed
using DOTS. The telemetry systems implements a DOTS client while the
orchestrator implements a DOTS server.

The communication between a network administrator and the
orchestrator is also performed using DOTS. The network administrator
via its web interfaces implements a DOTS client, while the
Orchestrator implements a DOTS server.

The communication between the Orchestrator and the DDoS mitigation
systems is performed using DOTS. The Orchestrator implements a DOTS
client while the DDoS mitigation systems implement a DOTS server.

The configuration aspects of each DDoS mitigation system, as well as
the instantiations of DDoS mitigation functions or network

configuration is not part of DOTS. Similarly, the discovery of
available DDoS mitigation functions is not part of DOTS.

```
                    +----------+
                    │ network  │C
                    │ adminis  │<-+
                    │ trator   │  │
                    +----------+  │
                                  │                (internal)
    +----------+  │ S+-------------+     +----------+
    │telemetry/│  +->│             │C   S│ DDoS     │+
    │monitoring│<--->│ Orchestrator│<--->│ mitigation│ │
    │systems   │  │  │             │<-+  │ systems  │ │
    +----------+C  S +-------------+C │   +----------+│
                                     │     +----------+
                                     │
                                     │    (external)
                                     │   +----------+
                                     │ S │ DDoS     │
                                     +->│ mitigation│
                                         │ systems  │
                                         +----------+
```

          * C is for DOTS client functionality
          * S is for DOTS server functionality

      Figure 1: DDoS Orchestration

The telemetry systems monitor various traffic network and perform
their measurement tasks. They are configured so that when an event
or some measurements reach a predefined level to report a DOTS
mitigation request to the Orchestrator. The DOTS mitigation request
may be associated with some element such as specific reporting.

Upon receipt of the DOTS mitigation request from the telemetry
system, the Orchestrator responds with an acknowledgement, to avoid
retransmission of the request for mitigation. The status of the DDoS
mitigation indicates the Orchestrator is in an analysing phase. The
Orchestrator begins collecting various information from various
telemetry systems in order to correlate the measurements and provide
an analysis of the event. Eventually, the Orchestrator may ask
additional information to the telemetry system, however, the
collection of these information is performed outside DOTS.

The orchestrator may be configured to start a DDoS mitigation upon
approval from a network administrator. The analysis from the
orchestrator is reported to the network administrator via a web

interface. If the network administrator decides to start the
mitigation, she orders through her web interface a DOTS client to
send a request for DDoS mitigation. This request is expected to be
associated with a context that identifies the DDoS mitigation
selected.

Upon receiving the DOTS request for DDoS mitigation from the network
administrator, the orchestrator orchestrates the DDoS mitigation
according to the specified strategy. Its status indicates the DDoS
mitigation is starting while not effective.

Orchestration of the DDoS mitigation systems works similarly as
described in Section 3.1.1 and Section 3.1.4. The Orchestrator
indicates with its status whether the DDoS Mitigation is effective.

When the DDoS mitigation is finished on the DDoS mitigation systems,
the orchestrator indicates to the Telemetry systems as well as to the
network administrator the DDoS mitigation is finished.

4. Security Considerations

DOTS is at risk from three primary attacks: DOTS agent impersonation,
traffic injection, and signaling blocking. Associated security
requirements and additional ones are defined in
[I-D.ietf-dots-requirements].

Impersonation and traffic injection mitigation can be managed through
current secure communications best practices. DOTS is not subject to
anything new in this area. One consideration could be to minimize
the security technologies in use at any one time. The more needed,
the greater the risk of failures coming from assumptions on one
technology providing protection that it does not in the presence of
another technology.

5. IANA Considerations

No IANA considerations exist for this document at this time.

6. Acknowledgments

The authors would like to thank among others Tirumaleswar Reddy, ,
Andrew Mortensen, Mohamed Boucadaire, and the DOTS WG chairs Roman D.
Danyliw and Tobias Gondrom for their valuable feedback.

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>.

7.2. Informative References

   [I-D.ietf-dots-requirements]
              Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed
              Denial of Service (DDoS) Open Threat Signaling
              Requirements", draft-ietf-dots-requirements-06 (work in
              progress), July 2017.

   [I2NSF]   "Interface to Network Security Functions (i2nsf)", n.d.,
              <https://datatracker.ietf.org/wg/i2nsf/about/>.

   [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>.

   [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J.
              Weil, "IPv6 Home Networking Architecture Principles",
              RFC 7368, DOI 10.17487/RFC7368, October 2014,
              <https://www.rfc-editor.org/info/rfc7368>.

Authors' Addresses

   Roland Dobbins
   Arbor Networks
   Singapore

   EMail: rdobbins@arbor.net

   Daniel Migault
   Ericsson
   8400 boulevard Decarie
   Montreal, QC H4P 2N2
   Canada

   EMail: daniel.migault@ericsson.com

Stefan Fouant
USA

EMail: stefan.fouant@copperriverit.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
USA

EMail: rgm@labs.htt-consult.com

Nik Teague
Verisign
12061 Bluemont Way
Reston, VA 20190

EMail: nteague@verisign.com

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

EMail: Frank.xialiang@huawei.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

EMail: kaname@nttv6.jp

DOTS   R. Dobbins
Internet-Draft   Arbor Networks
Intended status: Informational   D. Migault
Expires: March 8, 2020   Ericsson
                                                       R. Moskowitz
                                                     HTT Consulting
                                                         N. Teague
                                         Iron Mountain Data Centers
                                                            L. Xia
                                                            Huawei
                                                      K. Nishizuka
                                               NTT Communications
                                               September 05, 2019

                 Use cases for DDoS Open Threat Signaling
                    draft-ietf-dots-use-cases-20

Abstract

   The DDoS Open Threat Signaling (DOTS) effort is intended to provide
   protocols to facilitate interoperability across disparate DDoS
   mitigation solutions. This document presents sample use cases which
   describe the interactions expected between the DOTS components as
   well as DOTS messaging exchanges. These use cases are meant to
   identify the interacting DOTS components, how they collaborate, and
   what are the typical information to be exchanged.

Copyright Notice

Table of Contents

1. Introduction

   At the time of writing, distributed denial-of-service (DDoS) attack
   mitigation solutions are largely based upon siloed, proprietary
   communications schemes with vendor lock-in as a side-effect. This
   can result in the configuration, provisioning, operation, and
   activation of these solutions being a highly manual and often time-
   consuming process. Additionally, coordinating multiple DDoS
   mitigation solutions simultaneously is fraught with both technical
   and process-related hurdles. This greatly increases operational
   complexity which, in turn, can degrade the efficacy of mitigations.

   The DDoS Open Threat Signaling (DOTS) effort is intended to specify
   protocols that facilitate interoperability between diverse DDoS
   mitigation solutions and ensure greater integration in term of attack
   detection, mitigation requests, and attack characterization patterns.
   As DDoS solutions are broadly heterogeneous among vendors, the

primary goal of DOTS is to provide high-level interaction amongst
differing DDoS solutions, such as detecting, initiating, terminating
DDoS mitigation assistance or requesting the status of a DDoS
mitigation.

This document provides sample use cases to provide inputs for the
design of the DOTS protocols. The use cases are not exhaustive and
future use cases are expected to emerge as DOTS is adopted and
evolves.

2. Terminology and Acronyms

This document makes use of the same terminology and definitions as
[RFC8612]. In addition it uses the terms defined below:

o DDoS Mitigation Service Provider: designates the administrative
   entity providing the DDoS Mitigation Service.

o DDoS Mitigation System (DMS): A system that performs DDoS
   mitigation. The DDoS Mitigation System may be composed by a
   cluster of hardware and/or software resources, but could also
   involve an orchestrator that may take decisions such as
   outsourcing partial or more of the mitigation to another DDoS
   Mitigation System.

o DDoS Mitigation: The action performed by the DDoS Mitigation
   System.

o DDoS Mitigation Service: designates a service provided to a
   customer to mitigate DDoS attacks. Service subscriptions usually
   involve Service Level Agreement (SLA) that have to be met. It is
   the responsibility of the DDoS Service provider to instantiate the
   DDoS Mitigation System to meet these SLAs.

o Internet Transit Provider (ITP): designates the entity that
   delivers the traffic to a customer network. It can be an Internet
   Service Provider (ISP), or an upstream entity delivering the
   traffic to the ISP.

3. Use Cases

3.1. Upstream DDoS Mitigation by an Upstream Internet Transit Provider

This use case describes how an enterprise or a residential customer
network may take advantage of a pre-existing relation with its
Internet Transit Provider (ITP) in order to mitigate a DDoS attack
targeting its network.

To improve the clarity of our purpose, the targeted network will be designated as enterprise network, but the same scenario applies to any downstream network, including residential network and cloud hosting network.

As the ITP provides connectivity to the enterprise network, it is already on the path of the inbound and outbound traffic of the enterprise network and well aware of the networking parameters associated to the enterprise network WAN connectivity. This eases both the configuration and the instantiation of a DDoS Mitigation Service.

This section considers two kind of DDoS Mitigation Service between an enterprise network and an ITP:

o The upstream ITP may instantiate a DDoS Mitigation System (DMS) upon receiving a request from the enterprise network. This typically corresponds to the case when the enterprise network is under attack.

o On the other hand, the ITP may identify an enterprise network as the source of an attack and send a mitigation request to the enterprise DMS to mitigate this at the source.

The two scenarios, thought different, have similar interactions between the DOTS client and server. For the sake of simplicity, only the first scenario will be detailed in this section. Nevertheless, the second scenario is also in scope of DOTS.

In the first scenario, as depicted in Figure 1, an enterprise network with self-hosted Internet-facing properties such as Web servers, authoritative DNS servers, and VoIP servers has a DMS deployed to protect those servers and applications from DDoS attacks. In addition to on-premise DDoS defense capability, the enterprise has contracted with its ITP for DDoS Mitigation Services which threaten to overwhelm their WAN link(s) bandwidth.

```
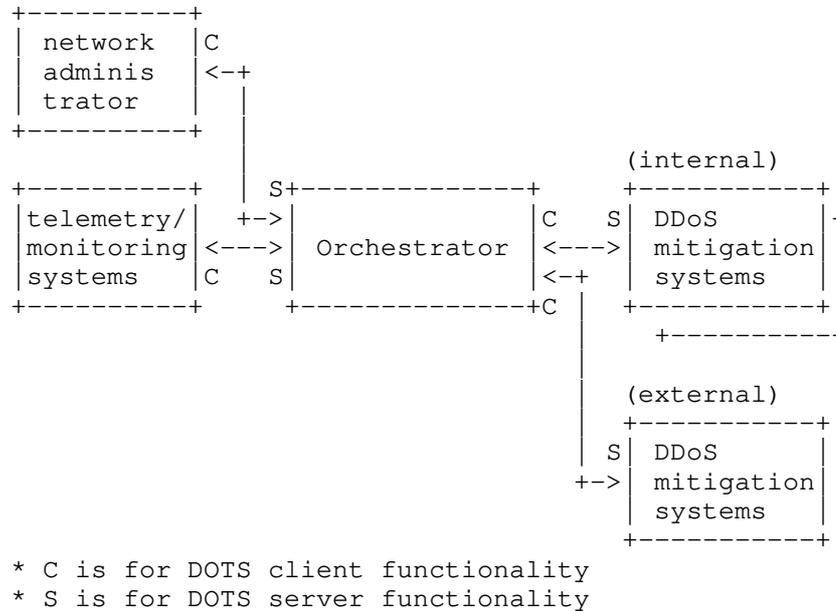+-----------------+        +-----------------+
| Enterprise      |        | Upstream        |
| Network         |        | Internet Transit|
|                 |        | Provider        |
|     +--------+  |        |          DDoS Attack
|     | DDoS   |  |  <==============================
|     | Target |  |  <==============================
|     +--------+  |        |   +-----------+  |
|                 |  +-------->| DDoS      |  |
|                 |  |     | S | Mitigation|  |
|                 |  |     |   | System    |  |
|                 |  |     |   +-----------+  |
|                 |  |     |                  |
|                 |  |     |                  |
|                 |  |     |                  |
|  +-----------+  |  |     |                  |
|  | DDoS      |<---+     |                  |
|  | Mitigation|C |        |                  |
|  | System    |  |        |                  |
|  +-----------+  |        |                  |
+-----------------+        +-----------------+
```

          * C is for DOTS client functionality
          * S is for DOTS server functionality

       Figure 1: Upstream Internet Transit Provider DDoS Mitigation

   The enterprise DMS is configured such that if the incoming Internet
   traffic volume exceeds 50% of the provisioned upstream Internet WAN
   link capacity, the DMS will request DDoS mitigation assistance from
   the upstream transit provider. More sophisticated detection means
   may be considered.

   The requests to trigger, manage, and finalize a DDoS Mitigation
   between the enterprise DMS and the ITP is performed using DOTS. The
   enterprise DMS implements a DOTS client while the ITP implements a
   DOTS server which is integrated with their DMS in this example.

   When the enterprise DMS locally detects an inbound DDoS attack
   targeting its resources (e.g., servers, hosts, or applications), it
   immediately begins a DDoS Mitigation.

   During the course of the attack, the inbound traffic volume to the
   enterprise network exceeds the 50% threshold and the enterprise DMS
   escalates the DDoS mitigation. The enterprise DMS DOTS client
   signals to the DOTS server on the upstream ITP to initiate DDoS
   Mitigation. The DOTS server replies to the DOTS client that it can

serve this request, and mitigation is initiated on the ITP network by
the ITP DMS.

Over the course of the attack, the DOTS server of the ITP
periodically informs the DOTS client on the enterprise DMS mitigation
status, statistics related to DDoS attack traffic mitigation, and
related information. Once the DDoS attack has ended, or decreased to
the certain level that the enterprise DMS can handle by itself, the
DOTS server signals the enterprise DMS DOTS client that the attack
has subsided.

The DOTS client on the enterprise DMS then requests the ITP to
terminate the DDoS Mitigation. The DOTS server on the ITP receives
this request and once the mitigation has ended, confirms the end of
upstream DDoS Mitigation to the enterprise DMS DOTS client.

The following is an overview of the DOTS communication model for this
use-case:

o (a) A DDoS attack is initiated against resources of a network
   organization (here, the enterprise) which has deployed a DOTS-
   capable DMS - typically a DOTS client.

o (b) The enterprise DMS detects, classifies, and begins the DDoS
   Mitigation.

o (c) The enterprise DMS determines that its capacity and/or
   capability to mitigate the DDoS attack is insufficient, and sends
   via its DOTS client a DOTS DDoS Mitigation request to one or more
   DOTS servers residing on the upstream ITP.

o (d) The DOTS server which receives the DOTS Mitigation request
   determines that it has been configured to honor requests from the
   requesting DOTS client, and honors its DDoS Mitigation by
   orchestrating its DMS.

o (e) While the DDoS Mitigation is active, the DOTS server regularly
   transmits DOTS DDoS Mitigation status updates to the DOTS client.

o (f) Informed by the DOTS server status update that the attack has
   ended or subsided, the DOTS client transmits a DOTS DDoS
   Mitigation termination request to the DOTS server.

o (g) The DOTS server terminates DDoS Mitigation, and sends the
   notification to the DOTS client.

Note that communications between the enterprise DOTS client and the
upstream ITP DOTS server may take place in-band within the main

Internet WAN link between the enterprise and the ITP; out-of-band via
a separate, dedicated wireline network link utilized solely for DOTS
signaling; or out-of-band via some other form of network connectivity
such as a third-party wireless 4G network connectivity.

Note also that a DOTS client that sends a DOTS Mitigation request may
be also triggered by a network admin that manually confirms the
request to the upstream ITP, in which case the request may be sent
from an application such as a web browser or a dedicated mobile
application.

Note also that when the enterprise is multihomed and connected to
multiple upstream ITPs, each ITP is only able to provide a DDoS
Mitigation Service for the traffic it transits. As a result, the
enterprise network may require to coordinate the various DDoS
Mitigation Services associated to each link. More multi-homing
considerations are discussed in [I-D.ietf-dots-multihoming].

3.2. DDoS Mitigation by a Third Party DDoS Mitigation Service Provider

This use case differs from the previous use case described in
Section 3.1 in that the DDoS Mitigation Service is not provided by an
upstream ITP. In other words, as represented in Figure 2, the
traffic is not forwarded through the DDoS Mitigation Service Provider
by default. In order to steer the traffic to the DDoS Mitigation
Service Provider, some network configuration changes are required.
As such, this use case is likely to match large enterprises or large
data centers, but not exclusively.

Another typical scenario for this use case is the relation between
DDoS Mitigation Service Providers forming an overlay of DMS. When a
DDoS Mitigation Service Provider mitigating a DDoS attack reaches it
resources capacities, it may chose to delegate the DDoS Mitigation to
another DDoS Mitigation Service Provider.

```
+-----------------+        +-----------------+
| Enterprise      |        | Upstream        |
| Network         |        | Internet Transit|
|                 |        | Provider        |
|                 |        |          DDoS Attack
|   +--------+    |        |                 |
|   | DDoS   |  <=============================
|   | Target |  <=============================
|   +--------+    |        |                 |
|                 |        |                 |
|                 |        +-----------------+
|                 |
|                 |        +-----------------+
|                 |        | DDoS Mitigation |
|                 |        | Service Provider|
|                 |        |                 |
|                 |        |                 |
| +-----------+   |        |  +-----------+  |
| | DDoS      |  <----------->|  DDoS      |  |
| | Mitigation|C |        |S |  Mitigation|  |
| | System    |   |        |  |  System    |  |
| +-----------+   |        |  +-----------+  |
+-----------------+        +-----------------+
```

```
          * C is for DOTS client functionality
          * S is for DOTS server functionality
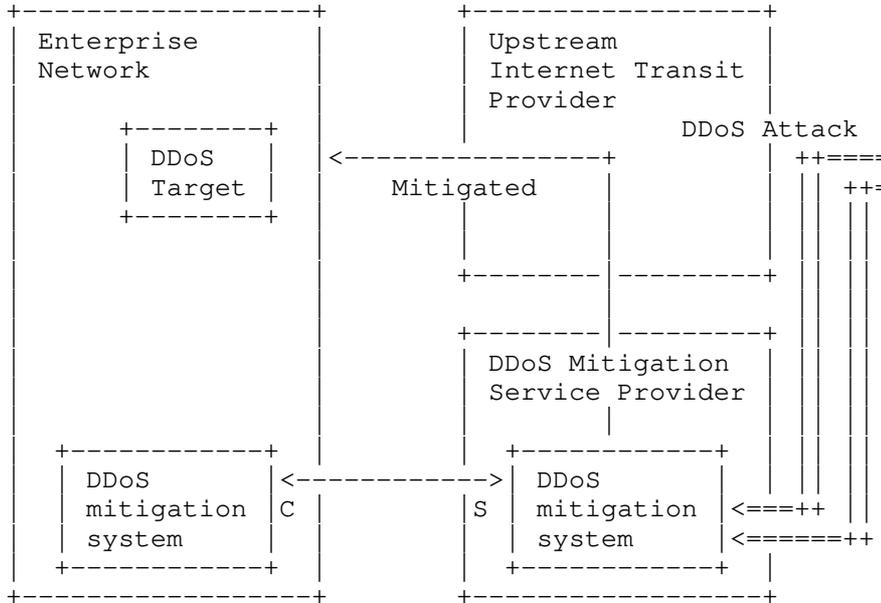```

          Figure 2: DDoS Mitigation between an Enterprise Network and Third
                 Party DDoS Mitigation Service Provider

   In this scenario, an enterprise network has entered into a pre-
   arranged DDoS mitigation assistance agreement with one or more other
   DDoS Mitigation Service Providers in order to ensure that sufficient
   DDoS mitigation capacity and/or capabilities may be activated in the
   event that a given DDoS attack threatens to overwhelm the ability of
   the enterprise's or any other given DMS to mitigate the attack on its
   own.

   The pre-arrangement typically includes the agreement on the
   mechanisms used to redirect the traffic to the DDoS Mitigation
   Service Provider, as well as the mechanism to re-inject the traffic
   back to the Enterprise Network.  Redirection to the DDoS Mitigation
   Service Provider typically involves BGP prefix announcement or DNS
   redirection, while re-injection of the scrubbed traffic to the
   enterprise network may be performed via tunneling mechanisms (e.g.,
   GRE).  These exact mechanisms used for traffic steering are out of
   scope.

   In some cases the communication between the enterprise DOTS client
   and the DOTS server of the DDoS Mitigation Service Provider may go

through the ITP carrying the DDoS attack, which would affect the
communication.  On the other hand, the communication between the DOTS
client and DOTS server may take a path that is not undergoing a DDoS
attack.

```
+-----------------+        +-----------------+
| Enterprise      |        | Upstream        |
| Network         |        | Internet Transit|
|                 |        | Provider        |
|                 |        |          DDoS Attack
|   +--------+    |        |           |  ++=====
|   | DDoS   |    <----------------+   |  ||  ++=
|   | Target |    | Mitigated  |   |   |  ||  ++=
|   +--------+    |        |        |   |  ||
|                 |        |        |   |  ||
|                 |        | +--------|---------+ ||
|                 |        | +--------|---------+ ||
|                 |        | | DDoS Mitigation | ||
|                 |        | | Service Provider| ||
|                 |        | |         |       | ||
|   +-----------+ |        | | +-----------+   | ||
|   | DDoS      | | <----------->| DDoS      |  | ||
|   | mitigation|C|        | |S | mitigation|  <===++ ||
|   | system    | |        | | | system    |  <======++
|   +-----------+ |        | | +-----------+   |
+-----------------+        +-----------------+
```

     * C is for DOTS client functionality
     * S is for DOTS server functionality

   Figure 3: Redirection to a DDoS Mitigation Service Provider

   When the enterprise network is under attack or at least is reaching
   its capacity or ability to mitigate a given DDoS attack traffic, the
   DOTS client sends a DOTS request to the DDoS Mitigation Service
   Provider to initiate network traffic diversion - as represented in
   Figure 3 - and DDoS mitigation activities. Ongoing attack and
   mitigation status messages may be passed between the enterprise
   network and the DDoS Mitigation Service Provider using DOTS. If the
   DDoS attack has stopped or the severity of the attack has subsided,
   the DOTS client can request the DDoS Mitigation Service Provider to
   stop the DDoS Mitigation.

3.3. DDoS Orchestration

   In this use case, one or more DDoS telemetry systems or monitoring
   devices monitor a network - typically an ISP network, an enterprise
   network, or a data center. Upon detection of a DDoS attack, these

DDoS telemetry systems alert an orchestrator in charge of
coordinating the various DMS's within the domain. The DDoS telemetry
systems may be configured to provide required information, such as a
preliminary analysis of the observation to the orchestrator.

The orchestrator analyses the various information it receives from
DDoS telemetry system, and initiates one or multiple DDoS mitigation
strategies. For example, the orchestrator could select the DDoS
mitigation system in the enterprise network or one provided by the
ITP.

DDoS Mitigation System selection and DDoS Mitigation techniques may
depends on the type of the DDoS attack. In some case, a manual
confirmation or selection may also be required to choose a proposed
strategy to initiate a DDoS Mitigation. The DDoS Mitigation may
consist of multiple steps such as configuring the network, or
updating already instantiated DDoS mitigation functions. Eventually,
the coordination of the mitigation may involve external DDoS
mitigation resources such as a transit provider or a Third Party DDoS
Mitigation Service Provider.

The communication used to trigger a DDoS Mitigation between the DDoS
telemetry and monitoring systems and the orchestrator is performed
using DOTS. The DDoS telemetry system implements a DOTS client while
the orchestrator implements a DOTS server.

The communication between a network administrator and the
orchestrator is also performed using DOTS. The network administrator
uses a web interface which interacts with a DOTS client, while the
orchestrator implements a DOTS server.

The communication between the orchestrator and the DDoS Mitigation
Systems is performed using DOTS. The orchestrator implements a DOTS
client while the DDoS Mitigation Systems implement a DOTS server.

The configuration aspects of each DDoS Mitigation System, as well as
the instantiations of DDoS mitigation functions or network
configuration is not part of DOTS. Similarly, the discovery of
available DDoS mitigation functions is not part of DOTS; and as such
is out of scope.

```
        +----------+
        | network  |C                (Enterprise Network)
        | adminis  |<-+
        | trator   |  |
        +----------+  |
                      |
        +----------+  | S+-------------+      +-----------+
        |telemetry/|  +->|             |C    S| DDoS      |+
        |monitoring|<--->| Orchestrator |<--->| mitigation||
        |systems   |C   S|             |<-+   | systems   ||
        +----------+     +-------------+C |   +-----------+|
                                          |     +----------+
        --------------------------------- |-----------------
                                          |
                                          |
           (Internet Transit Provider)    |
                                          |   +-----------+
                                          | S| DDoS      |+
                                          +->| mitigation||
                                          |  | systems   ||
                                          |  +-----------+|
        * C is for DOTS client functionality  +----------+
        * S is for DOTS server functionality
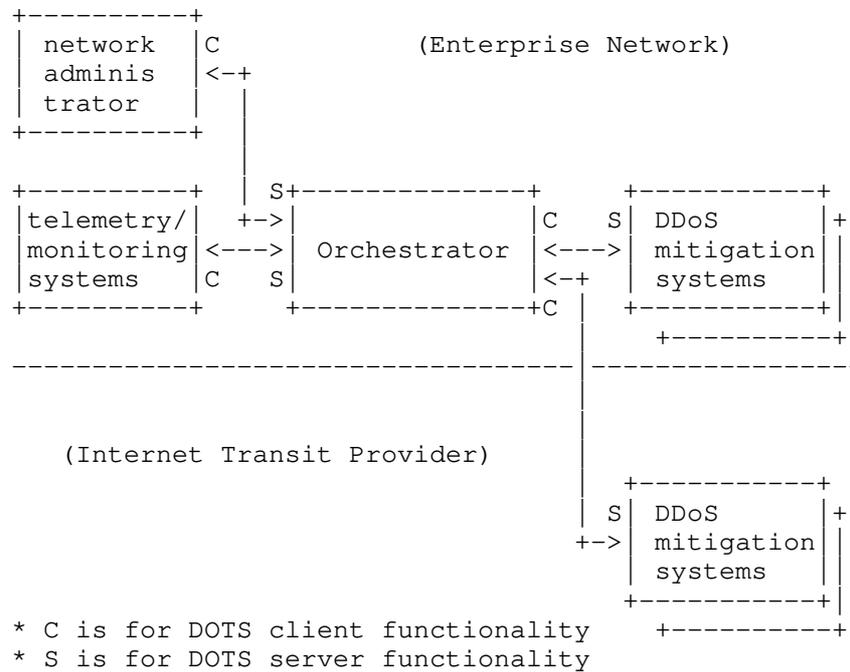```

           Figure 4: DDoS Orchestration

   The DDoS telemetry systems monitor various network traffic and
   perform some measurement tasks.

   These systems are configured so that when an event or some
   measurement indicators reach a predefined level their associated DOTS
   client sends a DOTS mitigation request to the orchestrator DOTS
   server.  The DOTS mitigation request may be associated with some
   optional mitigation hints to let the orchestrator know what has
   triggered the request.

   Upon receipt of the DOTS mitigation request from the DDoS telemetry
   system, the orchestrator DOTS server responds with an acknowledgment,
   to avoid retransmission of the request for mitigation.  The
   orchestrator may begin collecting additional fine-grained and
   specific information from various DDoS telemetry systems in order to
   correlate the measurements and provide an analysis of the event.
   Eventually, the orchestrator may ask for additional information from
   the DDoS telemetry system; however, the collection of this
   information is out of scope.

   The orchestrator may be configured to start a DDoS Mitigation upon
   approval from a network administrator.  The analysis from the

orchestrator is reported to the network administrator via a web
interface.  If the network administrator decides to start the
mitigation, the network administrator triggers the DDoS mitigation
request using the web interface of a DOTS client communicating to the
orchestrator DOTS server.  This request is expected to be associated
with a context that provides sufficient information to the
orchestrator DOTS server to infer the DDoS Mitigation to elaborate
and coordinate.

Upon receiving a request to mitigate a DDoS attack performed over a
target, the orchestrator may evaluate the volumetry of the attack as
well as the value that the target represents.  The orchestrator may
select the DDoS Mitigation Service Provider based on the attack
severity.  It may also coordinate the DDoS Mitigation performed by
the DDoS Mitigation Service Provider with some other tasks such as
for example, moving the target to another network so new sessions
will not be impacted.  The orchestrator requests a DDoS Mitigation to
the selected DDoS mitigation systems via its DOTS client, as
described in Section 3.1.

The orchestrator DOTS client is notified that the DDoS Mitigation is
effective by the selected DDoS mitigation systems.  The orchestrator
DOTS servers returns back this information to the network
administrator.

Similarly, when the DDoS attack has stopped, the orchestrator DOTS
client are being notified and the orchestrator's DOTS servers
indicate to the DDoS telemetry systems as well as to the network
administrator the end of the DDoS Mitigation.

In addition to the above DDoS Orchestration, the selected DDoS
mitigation system can return back a mitigation request to the
orchestrator as an offloading.  For example, when the DDoS attack
becomes severe and the DDoS mitigation system's utilization rate
reaches its maximum capacity, the DDoS mitigation system can send
mitigation requests with additional hints such as its blocked traffic
information to the orchestrator.  Then the orchestrator can take
further actions like requesting forwarding nodes such as routers to
filter the traffic.  In this case, the DDoS mitigation system
implements a DOTS client while the orchestrator implements a DOTS
server.  Similar to other DOTS use cases, the offloading scenario
assumes that some validation checks are followed by the DMS, the
orchestrator, or both (e.g., avoid exhausting the resources of the
forwarding nodes or disrupting the service).  These validation checks
are part of the mitigation, and are therefore out of the scope of the
document.

4.  Security Considerations

   The document does not describe any protocol.

   DOTS is at risk from three primary attacks: DOTS agent impersonation,
   traffic injection, and signaling blocking.

   Impersonation and traffic injection mitigation can be mitigated
   through current secure communications best practices.  Preconfigured
   mitigation steps to take on the loss of keepalive traffic can
   partially mitigate signal blocking, but in general it is impossible
   to comprehensively defend against an attacker that can selectively
   block any or all traffic

   Additional details of DOTS security requirements can be found in
   [RFC8612].

   Service disruption may be experienced if inadequate mitigation
   actions are applied.  These considerations are out of the scope of
   DOTS.

5.  IANA Considerations

   No IANA considerations exist for this document.

6.  Acknowledgments

   The authors would like to thank among others Tirumaleswar Reddy;
   Andrew Mortensen; Mohamed Boucadair; Artyom Gavrichenkov; Jon
   Shallow, Yuuhei Hayashi, the DOTS WG chairs, Roman Danyliw and Tobias
   Gondrom as well as the Security AD Benjamin Kaduk for their valuable
   feedback.

   We also would like to thank Stephan Fouant that was part of the
   initial co-authors of the documents.

7.  Informative References

   [I-D.ietf-dots-multihoming]
              Boucadair, M., K, R., and W. Pan, "Multi-homing Deployment
              Considerations for Distributed-Denial-of-Service Open
              Threat Signaling (DOTS)", draft-ietf-dots-multihoming-02
              (work in progress), July 2019.

   [RFC8612]  Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
              Threat Signaling (DOTS) Requirements", RFC 8612,
              DOI 10.17487/RFC8612, May 2019,
              <https://www.rfc-editor.org/info/rfc8612>.

Authors' Addresses

   Roland Dobbins
   Arbor Networks
   Singapore

   EMail: rdobbins@arbor.net


   Daniel Migault
   Ericsson
   8275 Trans Canada Route
   Saint Laurent, QC  4S 0B6
   Canada

   EMail: daniel.migault@ericsson.com


   Robert Moskowitz
   HTT Consulting
   Oak Park, MI  48237
   USA

   EMail: rgm@labs.htt-consult.com


   Nik Teague
   Iron Mountain Data Centers
   UK

   EMail: nteague@ironmountain.co.uk


   Liang Xia
   Huawei
   No. 101, Software Avenue, Yuhuatai District
   Nanjing
   China

   EMail: Frank.xialiang@huawei.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo  108-8118
Japan

EMail: kaname@nttv6.jp