DOTS
Internet-Draft
Intended status: Standards Track
Expires: May 16, 2018

T. Reddy
McAfee
M. Boucadair
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.

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.

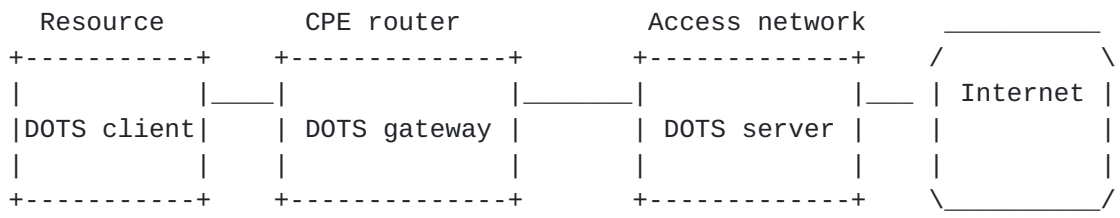

Figure 1: Sample DOTS Deployment (1)

The DOTS server can also be running on the Internet, as depicted in Figure 2.

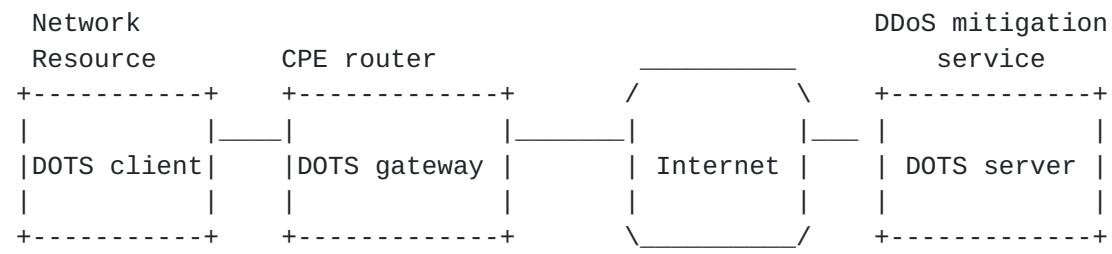

Figure 2: Sample DOTS Deployment (2)

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).

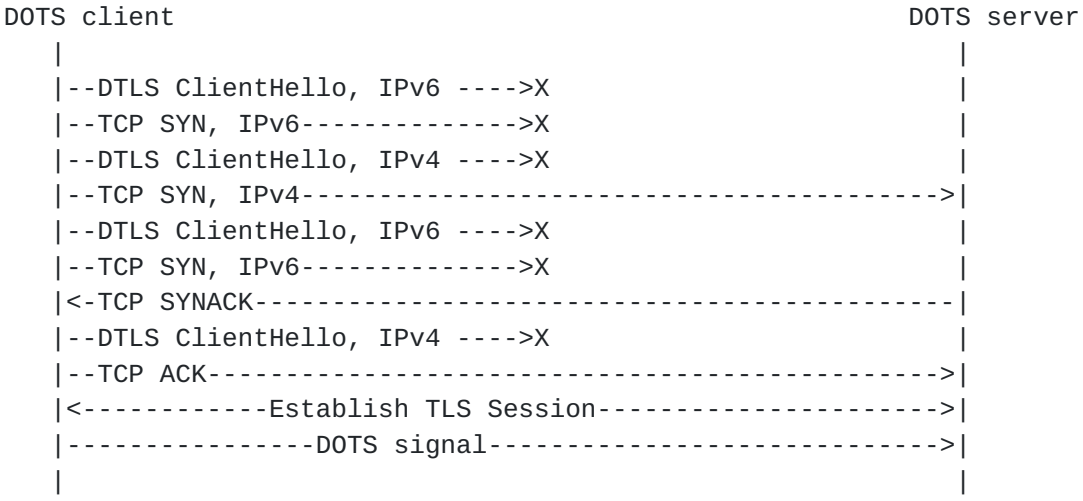

Figure 3: Happy Eyeballs

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.

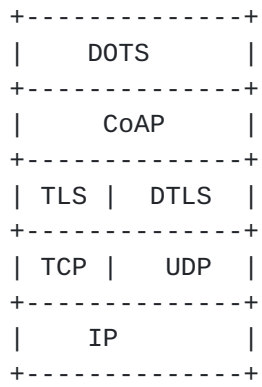

Figure 4: Abstract Layering of DOTS signal channel over CoAP over (D)TLS

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).

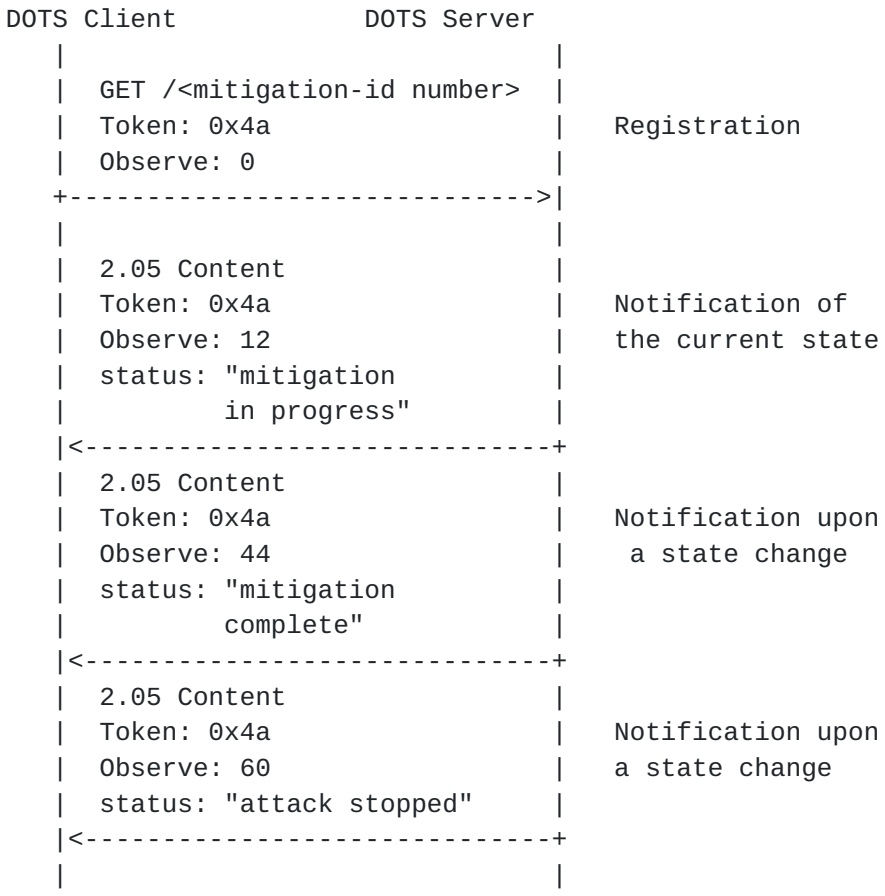

Figure 12: Notifications of attack mitigation status

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.

- 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.

- If the request is missing one or more mandatory attributes, then 4.00 (Bad Request) is returned in the response.

- If the request contains one or more invalid or unknown parameters, then 4.02 (Invalid query) code is returned in the response.

- 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:

- 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.

- 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:

- DOTS agents MUST support DTLS record replay detection (Section 3.3 of [RFC6347]) to protect against replay attacks.

- DOTS client can use (D)TLS session resumption without server-side state [RFC5077] to resume session and convey the DOTS signal.

- 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:

- TLS False Start [RFC7918] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.

- 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.

- 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:

- 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.

- 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.

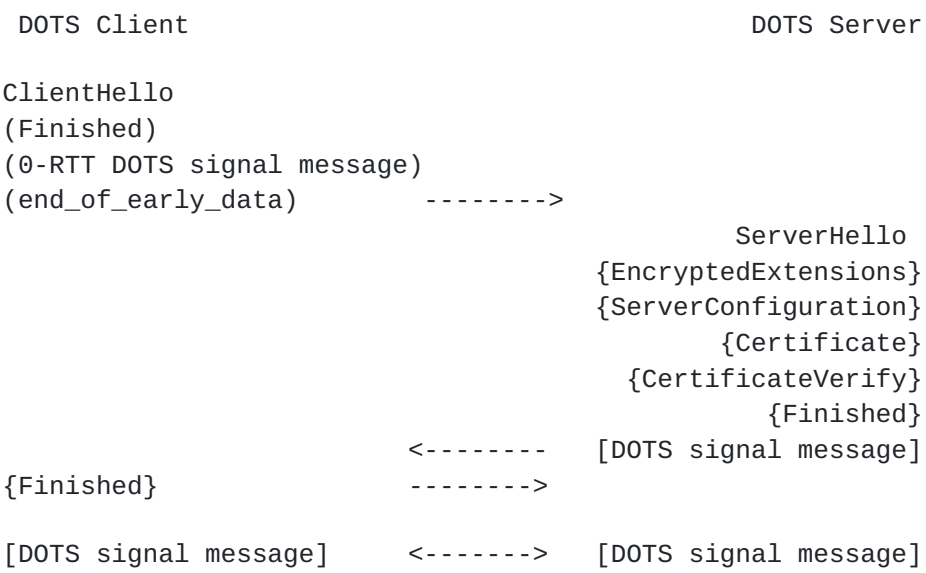

Figure 23: TLS 1.3 handshake with 0-RTT

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.

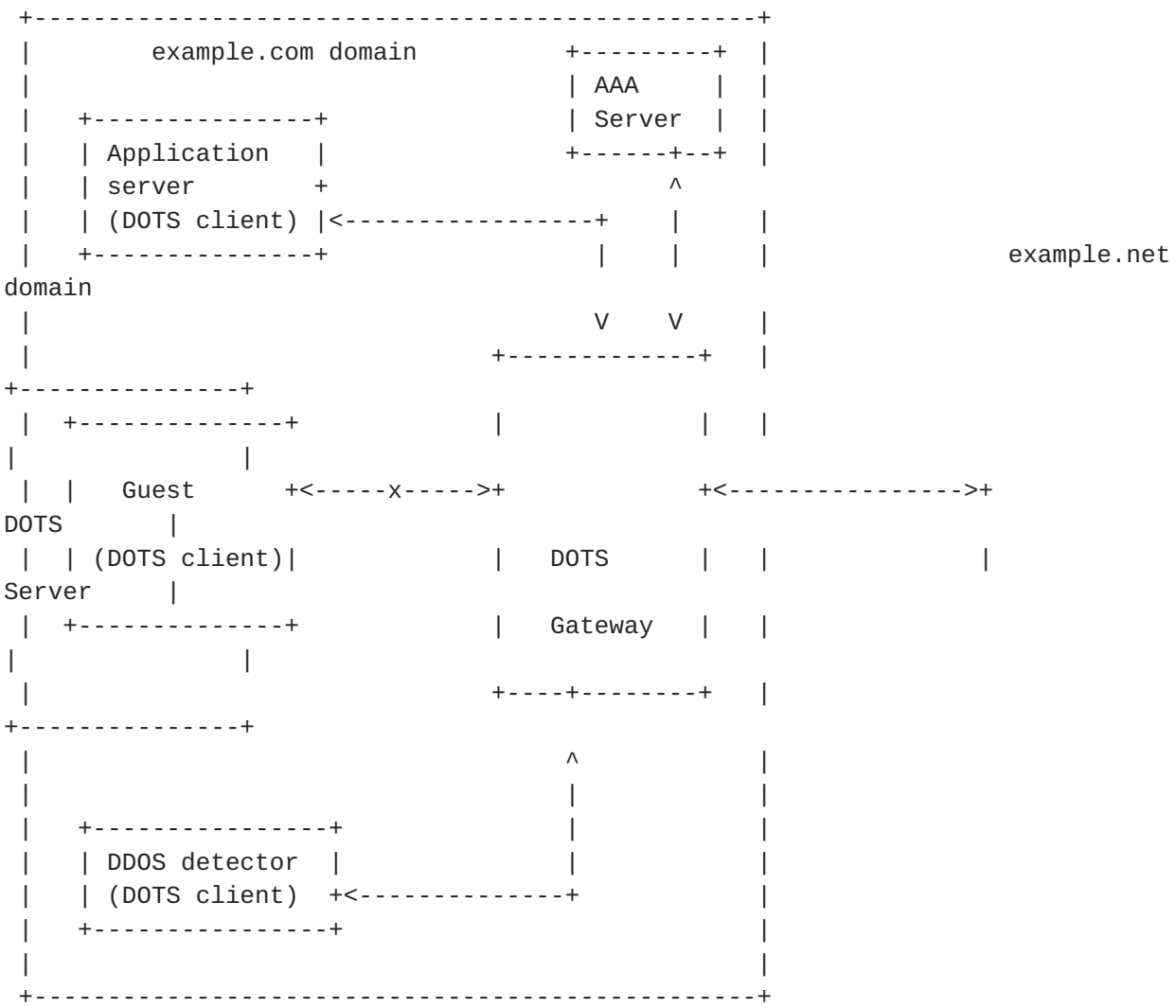

Figure 24: Example of Authentication and Authorization of DOTS Agents

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

- Parameter Name: "mitigation-scope"
- CBOR Key Value: 1
- CBOR Major Type: 5
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "scope"
- CBOR Key Value: 2
- CBOR Major Type: 5
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "mitigation-id"
- CBOR Key Value: 3
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:target-ip
- CBOR Key Value: 4
- CBOR Major Type: 4
- Change Controller: IESG

- Specification Document(s): this document

- Parameter Name: target-port-range
- CBOR Key Value: 5
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "lower-port"
- CBOR Key Value: 6
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "upper-port"
- CBOR Key Value: 7
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: target-protocol
- CBOR Key Value: 8
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "fqdn"
- CBOR Key Value: 9
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "uri"
- CBOR Key Value: 10
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: alias-name
- CBOR Key Value: 11
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: "lifetime"
- CBOR Key Value: 12
- CBOR Major Type: 0
- Change Controller: IESG

- Specification Document(s): this document

- Parameter Name: attack-status
- CBOR Key Value: 13
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: signal-config
- CBOR Key Value: 14
- CBOR Major Type: 5
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: heartbeat-interval
- CBOR Key Value: 15
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: max-retransmit
- CBOR Key Value: 16
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: ack-timeout
- CBOR Key Value: 17
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: ack-random-factor
- CBOR Key Value: 18
- CBOR Major Type: 7
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: MinValue
- CBOR Key Value: 19
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: MaxValue
- CBOR Key Value: 20
- CBOR Major Type: 0
- Change Controller: IESG

- Specification Document(s): this document

- Parameter Name: status
- CBOR Key Value: 21
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: bytes-dropped
- CBOR Key Value: 22
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: bps-dropped
- CBOR Key Value: 23
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: pkts-dropped
- CBOR Key Value: 24
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: pps-dropped
- CBOR Key Value: 25
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: session-id
- CBOR Key Value: 26
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: trigger-mitigation
- CBOR Key Value: 27
- CBOR Major Type: 7
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name: missing-hb-allowed
- CBOR Key Value: 28
- CBOR Major Type: 0
- Change Controller: IESG

- Specification Document(s): this document

- Parameter Name: CurrentValue
- CBOR Key Value: 29
- CBOR Major Type: 0
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:mitigation-start
- CBOR Key Value: 30
- CBOR Major Type: 7
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:target-prefix
- CBOR Key Value: 31
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:client-identifier
- CBOR Key Value: 32
- CBOR Major Type: 2
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:alt-server
- CBOR Key Value: 33
- CBOR Major Type: 2
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:alt-server-record
- CBOR Key Value: 34
- CBOR Major Type: 4
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:addr
- CBOR Key Value: 35
- CBOR Major Type: 2
- Change Controller: IESG
- Specification Document(s): this document

- Parameter Name:ttl
- CBOR Key Value: 36
- CBOR Major Type: 0
- 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