DOTS T. Reddy, Ed.
Internet-Draft McAfee
Intended status: Standards Track M. Boucadair, Ed.
Expires: August 2, 2018 Orange
K. Nishizuka
NTT Communications
L. Xia
Huawei
P. Patil
Cisco
A. Mortensen
Arbor Networks, Inc.
N. Teague
Verisign, Inc.
January 29, 2018

Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification

draft-ietf-dots-data-channel-13

Abstract

The document specifies a Distributed Denial-of-Service Open Threat Signaling (DOTS) data channel used for bulk exchange of data that cannot easily or appropriately communicated through the DOTS signal channel under attack conditions.

This is a companion document to the DOTS signal channel specification.

Editorial Note (To be removed by RFC Editor)

Please update these statements with the RFC number to be assigned to this document:

- "This version of this YANG module is part of RFC XXXX;"
- "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification";
- reference: RFC XXXX

Please update this statement with the RFC number to be assigned to the following documents:

- "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification";

o "RFC ZZZZ: Network Access Control List (ACL) YANG Data Model";

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale can be achieved by compromising enough end-hosts and using those infected hosts to perpetrate and amplify the attack. The victim of such attack can be an application server, a router, a firewall, an entire network, etc.

As discussed in [I-D.ietf-dots-requirements], the lack of a common method to coordinate a real-time response among involved actors and network domains inhibits the speed and effectiveness of DDoS attack mitigation. From that standpoint, DDoS Open Threat Signaling (DOTS) defines an architecture that allows a DOTS client to send requests to a DOTS server for DDoS attack mitigation [I-D.ietf-dots-architecture]. The DOTS approach is thus meant to minimize the impact of DDoS attacks, thereby contributing to the enforcement of more efficient defensive if not proactive security strategies. To that aim, DOTS defines two channels: the signal and the data channels (Figure 1).

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

Figure 1: DOTS Channels

The DOTS signal channel is used to carry information about a device or a network (or a part thereof) that is under a DDoS attack. Such information is sent by a DOTS client to an upstream DOTS server so that appropriate mitigation actions are undertaken on traffic deemed suspicious. The DOTS signal channel is further elaborated in [I-D.ietf-dots-signal-channel].

As for the DOTS data channel, it is used for infrequent bulk data exchange between DOTS agents to significantly improve the coordination of all the parties involved in the response to the attack. Section 2 of [I-D.ietf-dots-architecture] mentions that the DOTS data channel is used to perform the following tasks:

o Creating aliases for resources for which mitigation may be requested.

  A DOTS client may submit to its DOTS server a collection of prefixes which it would like to refer to by an alias when requesting mitigation. The DOTS server can respond to this request with either a success or failure response (see Section 2 in [I-D.ietf-dots-architecture]).

  Refer to Section 7 for more details.

o Filter management, which enables a DOTS client to request the installation or withdrawal of traffic filters, dropping or rate-limiting unwanted traffic, and permitting white-listed traffic. A DOTS client is entitled to instruct filtering rules only on IP resources that belong to its domain.

  Sample use cases for populating black- or white-list filtering rules are detailed hereafter:

  * If a network resource (DOTS client) detects a potential DDoS attack from a set of IP addresses, the DOTS client informs its servicing DOTS gateway of all suspect IP addresses that need to be blocked or black-listed for further investigation. The DOTS client could also specify a list of protocols and port numbers in the black-list rule.

    The DOTS gateway then propagates the black-listed IP addresses to a DOTS server which will undertake appropriate actions so that traffic originated by these IP addresses to the target network (specified by the DOTS client) is blocked.

  * A network, that has partner sites from which only legitimate traffic arrives, may want to ensure that the traffic from these sites is not subjected to DDoS attack mitigation. The DOTS client uses the DOTS data channel to convey the white-listed IP prefixes of the partner sites to its DOTS server.

    The DOTS server uses this information to white-list flows originated by such IP prefixes and which reach the network.

  Refer to Section 8 for more details.

2. Notational Conventions and Terminology

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

The reader should be familiar with the terms defined in [I-D.ietf-dots-requirements].

The terminology for describing YANG data modules is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [I-D.ietf-netmod-yang-tree-diagrams].

This document generalizes the notion of Access Control List (ACL) so that it is not device-specific [I-D.ietf-netmod-acl-model]. As such, this document defines an ACL as an ordered set of rules that is used to filter traffic. Each rule is represented by an Access Control Entry (ACE). ACLs communicated via the DOTS data channel are not bound to a device interface.

For the sake of simplicity, all of the examples in this document use "/restconf" as the discovered RESTCONF API root path. Many protocol header lines and message-body text within examples throughout the document are split into multiple lines for display purposes only. When a line ends with backslash ('\') as the last character, the line is wrapped for display purposes. It is to be considered to be joined to the next line by deleting the backslash, the following line break, and the leading whitespace of the next line.

3. DOTS Data Channel: Design Overview

Unlike the DOTS signal channel, which must remain operational even when confronted with signal degradation due to packets loss, the DOTS data channel is not expected to be fully operational at all times, especially when a DDoS attack is underway. The requirements for a DOTS data channel protocol are documented in [I-D.ietf-dots-requirements].

This specification does not require an order of DOTS signal and data channel creations nor mandates a time interval between them. These considerations are implementation- and deployment-specific.

As the primary function of the data channel is data exchange, a reliable transport mode is required in order for DOTS agents to detect data delivery success or failure. This document uses RESTCONF [RFC8040] over TLS [RFC5246] over TCP as the DOTS data channel protocol. The abstract layering of DOTS data channel is shown in Figure 2.

```
+-------------------+
| DOTS Data Channel |
+-------------------+
|     RESTCONF      |
+-------------------+
|       TLS         |
+-------------------+
|       TCP         |
+-------------------+
|       IP          |
+-------------------+
```

Figure 2: Abstract Layering of DOTS Data Channel

The HTTP POST, PUT, PATCH, and DELETE methods are used to edit data resources represented by DOTS data channel YANG data modules. These basic edit operations allow the DOTS data channel running configuration to be altered by a DOTS client.

DOTS data channel configuration information as well as state information can be retrieved with the GET method. An HTTP status-line header field is returned for each request to report success or failure for RESTCONF operations (Section 5.4 of [RFC8040]). The "error-tag" provides more information about encountered errors (Section 7 of [RFC8040]).

DOTS clients perform the root resource discovery procedure discussed in Section 3.1 of [RFC8040] to determine the root of the RESTCONF API. After discovering the RESTCONF API root, a DOTS client uses this value as the initial part of the path in the request URI, in any subsequent request to the DOTS server. The DOTS server may support the retrieval of the YANG modules it supports (Section 3.7 in [RFC8040]). For example, a DOTS client may use RESTCONF to retrieve the vendor-specific YANG modules supported by its DOTS server.

JavaScript Object Notation (JSON) [RFC7159] payload is used to propagate the DOTS data channel specific payload messages that carry request parameters and response information, such as errors. This specification uses the encoding rules defined in [RFC7951] for representing DOTS data channel configuration data using YANG (Section 5) as JSON text.

A DOTS client registers itself to its DOTS server(s) in order to set up DOTS data channel-related configuration data and receive state data (i.e., non-configuration data) from the DOTS server(s). Mutual authentication and coupling of signal and data channels are specified in [I-D.ietf-dots-signal-channel].

A single DOTS data channel between DOTS agents can be used to exchange multiple requests and multiple responses. To reduce DOTS client and DOTS server workload, DOTS clients SHOULD re-use the same TLS session. While the communication to the DOTS server is quiescent, the DOTS client MAY probe the server to ensure it has maintained cryptographic state. Such probes can also keep alive firewall and/or NAT bindings. A TLS heartbeat [RFC6520] verifies that the DOTS server still has TLS state by returning a TLS message.

In deployments where one or more translators (e.g., NAT44, NAT64, NPTv6) are enabled between the client's network and the DOTS server, DOTS data channel messages forwarded to a DOTS server must not include internal IP addresses/prefixes and/or port numbers; external addresses/prefixes and/or port numbers as assigned by the translator MUST be used instead. This document does not make any recommendation about possible translator discovery mechanisms. The following are some (non-exhaustive) deployment examples that may be considered:

o Port Control Protocol (PCP) [RFC6887] or Session Traversal Utilities for NAT (STUN) [RFC5389] may be used to retrieve the external addresses/prefixes and/or port numbers. Information retrieved by means of PCP or STUN will be used to feed the DOTS data channel messages that will be sent to a DOTS server.

o A DOTS gateway may be co-located with the translator. The DOTS gateway will need to update the DOTS messages, based upon the local translator's binding table.

When a server-domain DOTS gateway is involved in DOTS data channel exchanges, the same considerations for manipulating the 'cdid' (client domain identifier) parameter specified in [I-D.ietf-dots-signal-channel] MUST be followed by DOTS agents. As a reminder, 'cdid' is meant to assist the DOTS server to enforce some policies (e.g., limit the number of filtering rules per DOTS client or per DOTS client domain).

If a DOTS gateway is involved, the DOTS gateway verifies that the DOTS client is authorized to undertake a data channel action (e.g., instantiate filtering rules). If the DOTS client is authorized, it propagates the rules to the upstream DOTS server. Likewise, the DOTS server verifies that the DOTS gateway is authorized to relay data channel actions. For example, to create or purge filters, a DOTS client sends its request to its DOTS gateway. The DOTS gateway validates the rules in the request and proxies the requests containing the filtering rules to its DOTS server. When the DOTS gateway receives the associated response from the DOTS server, it propagates the response back to the DOTS client.

A DOTS server may detect conflicting filtering requests from distinct DOTS clients which belong to the same domain. For example, a DOTS client could request to blacklist a prefix by specifying the source prefix, while another DOTS client could request to whitelist that same source prefix, but both having the same destination prefix. It is out of scope of this specification to recommend the behavior to follow for handling conflicting requests (e.g., reject all, reject the new request, notify an administrator for validation). DOTS servers SHOULD support a configuration parameter to indicate the behavior to follow when a conflict is detected. Section 8.1 specifies the behavior when no instruction is supplied to a DOTS server.

How filtering rules instantiated on a DOTS server are translated into network configurations actions is out of scope.

4. DOTS Server(s) Discovery

This document assumes that DOTS clients are provisioned with the reachability information of their DOTS server(s) using a variety of means (e.g., local configuration, or dynamic means such as DHCP). The specification of such means are out of scope of this document.

Likewise, it is out of scope of this document to specify the behavior to follow by a DOTS client to place its requests (e.g., contact all servers, select one server among the list) when multiple DOTS servers are provisioned.

5. DOTS Data Channel YANG Module

5.1. Tree Structure

The tree structure for the DOTS data channel YANG module is as follows:

```
module: ietf-dots-data-channel
    +--rw dots-data
       +--rw dots-client* [cuid]
          +--rw cuid            string
          +--rw cdid?           string
          +--rw aliases
          |  +--rw alias* [name]
          |     +--rw name                 string
          |     +--rw target-prefix*       inet:ip-prefix
          |     +--rw target-port-range* [lower-port upper-port]
          |     |  +--rw lower-port    inet:port-number
          |     |  +--rw upper-port    inet:port-number
          |     +--rw target-protocol*     uint8
```

```
          |     +--rw target-fqdn*         inet:domain-name
          |     +--rw target-uri*          inet:uri
          |     +--rw lifetime             int32
          +--rw access-lists
             +--rw acl* [name]
                +--rw name        string
                +--rw type?       ietf-acl:acl-type
                +--rw lifetime    int32
                +--rw aces
                   +--rw ace* [name]
                      +--rw name          string
                      +--rw matches
                      |  +--rw (l3)?
                      |  |  +--:(ipv4)
                      |  |  |  +--rw ipv4
                      |  |  |     +--rw dscp?
                      |  |  |     |       inet:dscp
                      |  |  |     +--rw ecn?
                      |  |  |     |       uint8
                      |  |  |     +--rw length?
                      |  |  |     |       uint16
                      |  |  |     +--rw ttl?
                      |  |  |     |       uint8
                      |  |  |     +--rw protocol?
                      |  |  |     |       uint8
                      |  |  |     +--rw source-port-range-or-operator
                      |  |  |     |  +--rw (port-range-or-operator)?
                      |  |  |     |     +--:(range)
                      |  |  |     |     |  +--rw lower-port
                      |  |  |     |     |  |       inet:port-number
                      |  |  |     |     |  +--rw upper-port
                      |  |  |     |     |          inet:port-number
                      |  |  |     |     +--:(operator)
                      |  |  |     |        +--rw operator?
                      |  |  |     |        |       operator
                      |  |  |     |        +--rw port
                      |  |  |     |                inet:port-number
                      |  |  |     +--rw destination-port-range-or-operator
                      |  |  |     |  +--rw (port-range-or-operator)?
                      |  |  |     |     +--:(range)
                      |  |  |     |     |  +--rw lower-port
                      |  |  |     |     |  |       inet:port-number
                      |  |  |     |     |  +--rw upper-port
                      |  |  |     |     |          inet:port-number
                      |  |  |     |     +--:(operator)
                      |  |  |     |        +--rw operator?
                      |  |  |     |        |       operator
                      |  |  |     |        +--rw port
```

```
                         |  |  |     |                inet:port-number
                         |  |  |     +--rw ihl?
                         |  |  |     |       uint8
                         |  |  |     +--rw flags?
                         |  |  |     |       bits
                         |  |  |     +--rw offset?
                         |  |  |     |       uint16
                         |  |  |     +--rw identification?
                         |  |  |     |       uint16
                         |  |  |     +--rw destination-ipv4-network?
                         |  |  |     |       inet:ipv4-prefix
                         |  |  |     +--rw source-ipv4-network?
                         |  |  |     |       inet:ipv4-prefix
                         |  |  |     +--rw v4-fragments?
                         |  |  |             empty
                         |  |  +--:(ipv6)
                         |  |     +--rw ipv6
                         |  |        +--rw dscp?
                         |  |        |       inet:dscp
                         |  |        +--rw ecn?
                         |  |        |       uint8
                         |  |        +--rw length?
                         |  |        |       uint16
                         |  |        +--rw ttl?
                         |  |        |       uint8
                         |  |        +--rw protocol?
                         |  |        |       uint8
                         |  |        +--rw source-port-range-or-operator
                         |  |        |  +--rw (port-range-or-operator)?
                         |  |        |     +--:(range)
                         |  |        |     |  +--rw lower-port
                         |  |        |     |  |       inet:port-number
                         |  |        |     |  +--rw upper-port
                         |  |        |     |          inet:port-number
                         |  |        |     +--:(operator)
                         |  |        |        +--rw operator?
                         |  |        |        |       operator
                         |  |        |        +--rw port
                         |  |        |                inet:port-number
                         |  |        +--rw destination-port-range-or-operator
                         |  |        |  +--rw (port-range-or-operator)?
                         |  |        |     +--:(range)
                         |  |        |     |  +--rw lower-port
                         |  |        |     |  |       inet:port-number
                         |  |        |     |  +--rw upper-port
                         |  |        |     |          inet:port-number
                         |  |        |     +--:(operator)
                         |  |        |        +--rw operator?
```

```
  |  |        |        |       operator
  |  |        |        +--rw port
  |  |        |                inet:port-number
  |  |        +--rw destination-ipv6-network?
  |  |        |       inet:ipv6-prefix
  |  |        +--rw source-ipv6-network?
  |  |        |       inet:ipv6-prefix
  |  |        +--rw flow-label?
  |  |        |       inet:ipv6-flow-label
  |  |        +--rw v6-fragments?
  |  |                empty
  |  +--rw (l4)?
  |     +--:(tcp)
  |     |  +--rw tcp
  |     |     +--rw sequence-number?
  |     |     |       uint32
  |     |     +--rw acknowledgement-number?
  |     |     |       uint32
  |     |     +--rw data-offset?
  |     |     |       uint8
  |     |     +--rw reserved?
  |     |     |       uint8
  |     |     +--rw flags?
  |     |     |       bits
  |     |     +--rw window-size?
  |     |     |       uint16
  |     |     +--rw urgent-pointer?
  |     |     |       uint16
  |     |     +--rw options?
  |     |             uint32
  |     +--:(udp)
  |     |  +--rw udp
  |     |     +--rw length?   uint16
  |     +--:(icmp)
  |        +--rw icmp
  |           +--rw type?             uint8
  |           +--rw code?             uint8
  |           +--rw rest-of-header?   uint32
  +--rw actions
  |  +--rw forwarding    identityref
  |  +--rw rate-limit?   decimal64
  +--ro statistics
     +--ro matched-packets?   yang:counter64
     +--ro matched-octets?    yang:counter64
```

The DOTS data channel YANG module (ietf-dots-data-channel) allows to manage aliases for resources for which mitigation may be requested.

Such aliases may be used in subsequent DOTS signal channel exchanges to refer more efficiently to the resources under attack.

Also, the module allows managing filtering rules. Examples of filtering management in a DOTS context include, but not limited to:

- Black-list management, which enables a DOTS client to inform a DOTS server about sources from which traffic should be discarded.

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

- Filter management, which enables a DOTS client to request the installation or withdrawal of traffic filters, dropping or rate-limiting unwanted traffic and permitting white-listed traffic.

Early versions of this document investigated to what extent augmenting 'ietf-access-control-list' meet DOTS requirements, but that design approach was abandoned because it does not support meeting many of DOTS requirements, e.g.,

- Retrieve a filtering entry (or all entries) created by a DOTS client.

- Delete a filtering entry that was instantiated by a DOTS client.

DOTS filtering entries (i.e., Access Control List (ACL)) mimic the structure specified in [I-D.ietf-netmod-acl-model]. Concretely, DOTS agents are assumed to manipulate an ordered list of ACLs; each ACL contains a separately ordered list of Access Control Entries (ACEs). Each ACE has a group of match and a group of action criteria.

The 'ietf-dots-data-channel' module reuses the packet fields module 'ietf-packet-fields' [I-D.ietf-netmod-acl-model] which defines matching on fields in the packet including IPv4, IPv6, and transport layer fields. DOTS implementations MUST support the following matching criteria: match based on the IP header (IPv4 and IPv6), match based on the transport header (TCP, UDP, and ICMP), and any combination thereof.

DOTS forwarding actions can be 'accept' (i.e., accept matching traffic) or 'drop' (i.e., drop matching traffic without sending any ICMP error message). Accepted traffic can be subject to rate limiting 'rate-limit'. Note that 'reject' action (i.e., drop matching traffic and send an ICMP error message to the source) is not supported in 'ietf-dots-data-channel' because it is not appropriate in the context of DDoS mitigation. Generating ICMP messages to

notify drops when mitigating a DDoS attack will exacerbate the DDoS attack. Furthermore, these ICMP messages will be used by an attacker as an explicit signal that the traffic is being blocked.

Filtering rules instructed by a DOTS client assumes a default direction: the destination is the DOTS client domain.

This document supports fragment filtering which adds an additional layer of protection against a DoS attack that uses non-initial fragments only. When there is only layer 3 information in the ACL entry and the fragments keyword is present, for non-initial fragments matching the ACE entry, the 'deny' or 'accept' action associated with the ACE entry will be enforced. For initial or non-fragment matching the ACE entry, the next ACE entry will be processed. When there is both layer 3 and layer 4 information in the ACE entry and the fragments keyword is present, the ACE action is conservative for both 'accept' and 'deny' actions. The actions are conservative to not accidentally deny a fragmented portion of a flow because the fragments do not contain sufficient information to match all of the filter attributes. In the 'deny' action case, instead of denying a non-initial fragment, the next ACE entry is processed. In the 'accept' case, it is assumed that the layer 4 information in the non-initial fragment, if available, matches the layer 4 information in the ACE entry.

Once all the ACE entries have been iterated though with no match, then all the following ACL's ACE entries are iterated through until the first match at which point the specified action is applied. If there is no match, then there is no action to be taken against the packet.

In order to avoid stale entries, a lifetime is associated with alias and filtering entries. DOTS clients include a suggested lifetime in the request, but it is up to the DOTS server to decide whether it honors that hint or it has to proceed as per its local policies.

5.2. YANG Module

```
<CODE BEGINS> file "ietf-dots-data-channel@2018-01-22.yang"

module ietf-dots-data-channel {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-data-channel";
  prefix data-channel;

  import ietf-access-control-list {
    prefix ietf-acl;
  }
```

```
    import ietf-packet-fields {
      prefix packet-fields;
    }
    import ietf-dots-signal-channel {
      prefix dots-signal;
    }

    organization
      "IETF DDoS Open Threat Signaling (DOTS) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/dots/>
       WG List:  <mailto:dots@ietf.org>

       Editor:  Konda, Tirumaleswar Reddy
                <mailto:TirumaleswarReddy_Konda@McAfee.com>

       Editor:  Mohamed Boucadair
                <mailto:mohamed.boucadair@orange.com>

       Author:  Kaname Nishizuka
                <mailto:kaname@nttv6.jp>

       Author:  Liang Xia
                <mailto:frank.xialiang@huawei.com>

       Author:  Prashanth Patil
                <mailto:praspati@cisco.com>

       Author:  Andrew Mortensen
                <mailto:amortensen@arbor.net>

       Author:  Nik Teague
                <mailto:nteague@verisign.com>

       Author:  Jon Shallow
                <mailto:jon.shallow@nccgroup.trust>";
    description
      "This module contains YANG definition for configuring
       aliases for resources and filtering rules using DOTS
       data channel.

       Copyright (c) 2018 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 2018-01-22 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: Distributed Denial-of-Service Open Threat
                  Signaling (DOTS) Data Channel Specification";
   }

   grouping aliases {
     description
       "Top level container for aliases";
     list alias {
       key "name";
       description
         "List of aliases";
       leaf name {
         type string;
         description
           "The name of the alias";
       }
       uses dots-signal:target;
       leaf lifetime {
         type int32;
         units "minutes";
         mandatory true;
         description
           "Indicates the lifetime of the alias entry.

            A lifetime of negative one (-1) indicates indefinite
            lifetime for the alias.";
       }
     }
   }

   grouping access-lists {
     description
       "Specifies the ordered set of Access Control Lists.";
     list acl {
       key "name";
       ordered-by user;
       description
         "An Access Control List (ACL) is an ordered list of
```

```
             Access List Entries (ACE). Each Access Control Entry has a
             list of match criteria and a list of actions.";
          leaf name {
            type string {
              length "1..64";
            }
            description
              "The name of access-list.";
            reference
               "RFC ZZZZ: Network Access Control List (ACL)
                          YANG Data Model";
          }
          leaf type {
            type ietf-acl:acl-type;
            description
              "Type of access control list. Indicates the primary intended
               type of match criteria (e.g., IPv4, IPv6) used in the list
               instance.";
            reference
               "RFC ZZZZ: Network Access Control List (ACL)
                          YANG Data Model";
          }
          leaf lifetime {
            type int32;
            units "minutes";
            mandatory true;
            description
              "Indicates the lifetime of the filtering rule

               A lifetime of negative one (-1) indicates indefinite
               lifetime for the filtering request.";
          }
          container aces {
            description
              "The access-list-entries container contains
               a list of ACEs.";
            list ace {
              key "name";
              ordered-by user;
              description
                "List of access list entries.";
              leaf name {
                type string {
                  length "1..64";
                }
                description
                  "A unique name identifying this Access List
                   Entry (ACE).";
```

```
            reference
              "RFC ZZZZ: Network Access Control List (ACL)
                         YANG Data Model";
          }
          container matches {
            description
              "The rules in this set determine what fields will be
               matched upon before any action is taken on them.

               If no matches are defined in a particular container,
               then any packet will match that container.

               If no matches are specified at all in an ACE, then any
               packet will match the ACE.";
            reference
               "RFC ZZZZ: Network Access Control List (ACL)
                          YANG Data Model";

            choice l3 {
              container ipv4 {
                when "derived-from(../../../../type," +
                     "'ietf-acl:ipv4-acl-type')";
                uses packet-fields:acl-ip-header-fields;
                uses packet-fields:acl-ipv4-header-fields;
                leaf v4-fragments {
                  type empty;
                  description
                    "Handle IPv4 fragments.";
                }
                description
                  "Rule set that matches IPv4 header.";
              }
              container ipv6 {
                when "derived-from(../../../../type," +
                     "'ietf-acl:ipv6-acl-type')";
                uses packet-fields:acl-ip-header-fields;
                uses packet-fields:acl-ipv6-header-fields;
                leaf v6-fragments {
                  type empty;
                  description
                    "Handle IPv6 fragments.";
                }
                description
                  "Rule set that matches IPv6 header.";
              }
              description
                "Either IPv4 or IPv6.";
            }
```

```
              choice l4 {
                container tcp {
                  uses packet-fields:acl-tcp-header-fields;
                  description
                    "Rule set that matches TCP header.";
                }
                container udp {
                  uses packet-fields:acl-udp-header-fields;
                  description
                    "Rule set that matches UDP header.";
                }
                container icmp {
                  uses packet-fields:acl-icmp-header-fields;
                  description
                    "Rule set that matches ICMP header.";
                }
                description
                  "Can be TCP, UDP, or ICMP";
              }
            }
            container actions {
              description
                "Definitions of action for this ACE.";
              leaf forwarding {
                type identityref {
                  base ietf-acl:forwarding-action;
                }
                mandatory true;
                description
                  "Specifies the forwarding action per ACE.";
                reference
                   "RFC ZZZZ: Network Access Control List (ACL)
                              YANG Data Model";
              }
              leaf rate-limit {
                when "../forwarding = 'ietf-acl:accept'" {
                  description
                    "rate-limit valid only when accept action is used";
                }
                type decimal64 {
                  fraction-digits 2;
                }
                description
                  "rate-limit traffic";
              }
            }
            container statistics {
              config false;
```

```
              description
                "Aggregate statistics.";
              uses ietf-acl:acl-counters;
            }
          }
        }
      }
    }

    container dots-data {
      description
        "Main container for DOTS data channel.";
      list dots-client {
        key "cuid";
        description
          "List of DOTS clients.";
        leaf cuid {
          type string;
          description
            "A unique identifier that is randomly generated by
             a DOTS client to prevent request collisions.";
          reference
            "RFC YYYY: Distributed Denial-of-Service Open Threat
                     Signaling (DOTS) Signal Channel Specification";
        }
        leaf cdid {
          type string;
          description
            "A client domain identifier conveyed by a
             server-domain DOTS gateway to a remote DOTS server.";
          reference
            "RFC YYYY: Distributed Denial-of-Service Open Threat
                     Signaling (DOTS) Signal Channel Specification";
        }
        container aliases {
          description
            "Set of aliases that are bound to a DOTS client.";
          uses aliases;
        }
        container access-lists {
          description
            "Access lists that are bound to a DOTS client.";
           uses access-lists;
        }
      }
    }
  }
   <CODE ENDS>
```

6. Registering DOTS Clients

In order to create a new DOTS client ('dots-client') resource on DOTS servers, DOTS clients MUST send a POST request (shown in Figure 3).

```
POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
Host: {host}:{port}
Content-Type: application/yang-data+json
{
  "ietf-dots-data-channel:dots-client": [
    {
      "cuid": "string"
    }
  ]
}
```

Figure 3: POST to Register

The 'cuid' (client unique identifier) parameter is described below:

cuid: A globally unique identifier that is meant to prevent collisions among DOTS clients. This attribute has the same meaning, syntax, and processing rules as the 'cuid' attribute defined in [I-D.ietf-dots-signal-channel].

DOTS clients MUST use the same 'cuid' for both signal and data channels.
This is a mandatory attribute.

In deployments where server-domain DOTS gateways are enabled, identity information about the origin source client domain SHOULD be supplied to the DOTS server. That information is meant to assist the DOTS server to enforce some policies. These policies can be enforced per-client, per-client domain, or both. Figure 4 shows an example of a request relayed by a server-domain DOTS gateway.

```
POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
Host: {host}:{port}
Content-Type: application/yang-data+json
{
  "ietf-dots-data-channel:dots-client": [
    {
      "cuid": "string",
      "cdid": "string"
    }
  ]
}
```

Figure 4: POST to Register (DOTS Gateway)

A server-domain DOTS gateway SHOULD add the following attribute:

cdid: This attribute has the same meaning, syntax, and processing rules as the 'cdid' attribute defined in [I-D.ietf-dots-signal-channel].

The DOTS gateway that inserted a 'cdid' in a request, MUST strip the 'cdid' parameter in the corresponding response before forwarding the response to the DOTS client.

This is an optional attribute.

A request example to create a 'dots-client' resource is depicted in Figure 5. This request is relayed by a server-domain DOTS gateway as hinted by the presence of the 'cdid' attribute.

```
POST /restconf/data/ietf-dots-data-channel:dots-data HTTP/1.1
Host: {host}:{port}
Content-Type: application/yang-data+json
{
  "ietf-dots-data-channel:dots-client": [
    {
      "cuid": "dz6pHjaADkaFTbjr0JGBpw",
      "cdid": "7eeaf349529eb55ed50113"
    }
  ]
}
```

Figure 5: POST to Register (DOTS gateway)

DOTS servers MUST limit the number of 'dots-client' resources to be created by the same DOTS client to 1 per request. Requests with multiple 'dots-client' resources MUST be rejected by DOTS servers. To that aim, the DOTS server MUST rely on the same procedure to

unambiguously identify a DOTS client as discussed in Section 4.4.1 of [I-D.ietf-dots-signal-channel].

The DOTS server indicates the result of processing the POST request using status-line codes. Status codes in the range "2xx" codes are success, "4xx" codes are some sort of invalid requests and "5xx" codes are returned if the DOTS server has erred or is incapable of accepting the creation of the 'dots-client' resource. In particular,

o "201 Created" status-line is returned in the response, if the DOTS server has accepted the request.

o "400 Bad Request" status-line is returned by the DOTS server, if the request does not include a 'cuid' parameter. The error-tag "missing-attribute" is used in this case.

o "409 Conflict" status-line is returned to the requesting DOTS client, if the data resource already exists. The error-tag "resource-denied" is used in this case.

Once a DOTS client registers itself to a DOTS server, it can create/delete/retrieve aliases (Section 7) and filtering rules (Section 8).

A DOTS client MAY use the PUT request (Section 4.5 in [RFC8040]) to register a DOTS client within the DOTS server. An example is shown in Figure 6.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: {host}:{port}
Content-Type: application/yang-data+json
{
  "ietf-dots-data-channel:dots-client": [
    {
      "cuid": "dz6pHjaADkaFTbjr0JGBpw"
    }
  ]
}
```

Figure 6: PUT to Register

A DOTS client may de-register from its DOTS server by deleting the 'cuid' resource. An example is shown in Figure 7.

```
DELETE /restconf/data/ietf-dots-data-channel:dots-data\
       /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: {host}:{port}
```

Figure 7: De-register a DOTS Client

7. Managing DOTS Aliases

The following sub-sections define means for a DOTS client to create aliases (Section 7.1), retrieve one or a list of aliases (Section 7.2), and delete an alias (Section 7.3).

7.1. Create Aliases

A POST or PUT request is used by a DOTS client to create aliases, for resources for which a mitigation may be requested. Such aliases may be used in subsequent DOTS signal channel exchanges to refer more efficiently to the resources under attack.

DOTS clients within the same domain can create different aliases for the same resource.

The structure of POST requests used to create aliases is shown in Figure 8.

```
    POST /restconf/data/ietf-dots-data-channel:dots-data\
         /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
    Host: {host}:{port}
    Content-Type: application/yang-data+json
    {
     "ietf-dots-data-channel:aliases": {
       "alias": [
         {
           "name": "string",
           "target-prefix": [
             "string"
           ],
           "target-port-range": [
             {
               "lower-port": integer,
               "upper-port": integer
             }
           ],
           "target-protocol": [
             integer
           ],
           "target-fqdn": [
             "string"
           ],
           "target-uri": [
             "string"
           ],
           "lifetime": integer
         }
       ]
     }
   }
```

Figure 8: POST to Create Aliases

The parameters are described below:

name: Name of the alias.

This is a mandatory attribute.

target-prefix: Prefixes are separated by commas. Prefixes are represented using Classless Inter-domain Routing (CIDR) notation [RFC4632]. As a reminder, the prefix length must be less than or equal to 32 (resp. 128) for IPv4 (resp. IPv6).

The prefix list MUST NOT include broadcast, loopback, or multicast addresses. These addresses are considered as invalid values. In

addition, the DOTS server MUST validate that these prefixes are within the scope of the DOTS client's domain. Other validation checks may be supported by DOTS servers.

This is an optional attribute.

target-port-range: A range of port numbers.

The port range is defined by two bounds, a lower port number (lower-port) and an upper port number (upper-port).

When only 'lower-port' is present, it represents a single port number.

For TCP, UDP, Stream Control Transmission Protocol (SCTP) [RFC4960], or Datagram Congestion Control Protocol (DCCP) [RFC4340], the range of port numbers can be, for example, 1024-65535.

This is an optional attribute.

target-protocol: A list of protocols. Values are taken from the IANA protocol registry [proto_numbers].

The value '0' has a special meaning for 'all protocols'.

This is an optional attribute.

target-fqdn: A list of Fully Qualified Domain Names (FQDNs). An FQDN is the full name of a resource, rather than just its hostname. For example, "venera" is a hostname, and "venera.isi.edu" is an FQDN [RFC1983].

How a name is passed to an underlying name resolution library is implementation- and deployment-specific. Nevertheless, once the name is resolved into one or multiple IP addresses, DOTS servers MUST apply the same validation checks as those for 'target-prefix'.

This is an optional attribute.

target-uri: A list of Uniform Resource Identifiers (URIs) [RFC3986].

The same validation checks used for 'target-fqdn' MUST be followed by DOTS servers to validate a target URI.

This is an optional attribute.

lifetime: Lifetime of the alias, in minutes. The RECOMMENDED lifetime of an alias is 10080 minutes (1 week). DOTS clients MUST include this parameter in their alias creation requests. Upon the expiry of this lifetime, and if the request is not refreshed but no mitigation is active, the alias entry is removed.

The request can be refreshed by sending the same request again.

A lifetime of '0' in a request is an invalid value.

A lifetime of negative one (-1) indicates indefinite lifetime for the alias. The DOTS server MAY refuse indefinite lifetime, for policy reasons; the granted lifetime value is returned in the response. DOTS clients MUST be prepared to not be granted aliases with indefinite lifetimes.

The DOTS server MUST always indicate the actual lifetime in the response and the remaining lifetime in status messages sent to the DOTS client.

This is a mandatory attribute.

In POST requests, at least one of the 'target-prefix', 'target-fqdn', or 'target-uri' attributes MUST be present. DOTS agents can safely ignore Vendor-Specific parameters they don't understand.

Figure 9 shows a POST request to create an alias called "https1" for HTTPS servers with IP addresses 2001:db8:6401::1 and 2001:db8:6401::2 listening on port number 443.

```
POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: www.example.com
Content-Type: application/yang-data+json
{
  "ietf-dots-data-channel:aliases": {
    "alias": [
      {
        "name": "https1",
        "target-protocol": [
          6
        ],
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-port-range": [
          {
            "lower-port": 443
          }
        ],
        "lifetime": 10080
      }
    ]
  }
}
```

Figure 9: Example of a POST to Create an Alias

"201 Created" status-line MUST be returned in the response if the DOTS server has accepted the alias.

"409 Conflict" status-line MUST be returned to the requesting DOTS client, if the request is conflicting with an existing alias name. The error-tag "resource-denied" is used in this case.

If the request is missing a mandatory attribute or its contains an invalid or unknown parameter, "400 Bad Request" status-line MUST be returned by the DOTS server. The error-tag is set to "missing-attribute", "invalid-value", or "unknown-element" as a function of the encountered error.

A DOTS client MAY use the PUT request to modify the aliases in the DOTS server.

7.2. Retrieve Installed Aliases

GET request is used to retrieve one or all installed aliases by a DOTS client from a DOTS server (Section 3.3.1 in [RFC8040]). If no 'name' is included in the request, this is an indication that the request is about retrieving all aliases instantiated by the DOTS client.

Figure 10 shows an example to retrieve all the aliases that were instantiated by the requesting DOTS client. The 'content' parameter and its permitted values are defined in Section 4.8.1 of [RFC8040].

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /aliases?content=config HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 10: GET to Retrieve All Installed Aliases

Figure 11 shows an example of the response message body that includes all the aliases that are maintained by the DOTS server for the DOTS client identified by the 'cuid' parameter.

```
{
  "ietf-dots-data-channel:aliases": {
    "alias": [
      {
        "name": "Server1",
        "traffic-protocol": [
          6
        ],
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-port-range": [
          {
            "lower-port": 443
          }
        ],
        "lifetime": 3596
      },
      {
        "name": "Server2",
        "target-protocol": [
          6
        ],
        "target-prefix": [
          "2001:db8:6401::10/128",
          "2001:db8:6401::20/128"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          }
        ],
        "lifetime": 9869
      }
    ]
  }
}
```


Figure 11: An Example of Response Body

Figure 12 shows an example of a GET request to retrieve the alias "Server2" that was instantiated by the DOTS client.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /aliases/alias=Server2?content=config HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 12: GET to Retrieve an Alias

If an alias name ('name') is included in the request, but the DOTS server does not find that alias name for this DOTS client in its configuration data, it MUST respond with a "404 Not Found" status-line.

7.3. Delete Aliases

DELETE request is used to delete an alias maintained by a DOTS server.

If the DOTS server does not find the alias name, conveyed in the DELETE request, in its configuration data for this DOTS client, it MUST respond with a "404 Not Found" status-line.

The DOTS server successfully acknowledges a DOTS client's request to remove the alias using "204 No Content" status-line in the response.

Figure 13 shows an example of a request to delete an alias.

```
DELETE /restconf/data/ietf-dots-data-channel:dots-data\
       /dots-client=dz6pHjaADkaFTbjr0JGBpw\
       /aliases/alias=Server1 HTTP/1.1
Host: {host}:{port}
```

Figure 13: Delete an Alias

8. Managing DOTS Filtering Rules

The following sub-sections define means for a DOTS client to create filtering rules (Section 8.1), retrieve active filtering rules (Section 8.2), and delete a filtering rule (Section 8.3).

8.1. Install Filtering Rules

A POST or PUT request is used by a DOTS client to communicate filtering rules to a DOTS server.

Figure 14 shows a POST request example to block traffic from 192.0.2.0/24 and destined to 198.51.100.0/24.

```
    POST /restconf/data/ietf-dots-data-channel:dots-data\
         /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
    Host: {host}:{port}
    Content-Type: application/yang-data+json
    {
     "ietf-dots-data-channel:access-lists": {
       "acl": [
         {
           "name": "sample-ipv4-acl",
           "type": "ipv4-acl-type",
           "lifetime": 10080,
           "aces": {
             "ace": [
               {
                 "name": "rule1",
                 "matches": {
                   "l3": {
                     "ipv4" {
                        "destination-ipv4-network": "198.51.100.0/24"
                        "source-ipv4-network": "192.0.2.0/24",
                     }
                   }
                 },
                 "actions": {
                   "forwarding": "drop"
                 }
               }
             ]
           }
         }
       ]
     }
   }
```

Figure 14: POST to Install Filtering Rules

The meaning of these parameters is as follows:

name: The name of the access-list.

This is a mandatory attribute.

type: Indicates the primary intended type of match criteria (e.g., IPv4, IPv6). It is set to 'ipv4-acl-type' in this example.

This is an optional attribute.

lifetime: Lifetime of the ACL, in minutes. It MUST follow the same rules specified in Section 7.1 for alias lifetime, but applied to an ACL.

This is a mandatory attribute.

matches: Define criteria used to identify a flow on which to apply the rule. It can be "l3" (IPv4, IPv6) or "l4" (TCP, UDP, ..). The detailed match parameters are specified in Section 5.

In this example, an IPv4 matching criteria is used.

This is an optional attribute.

destination-ipv4-network: The destination IPv4 prefix. DOTS servers MUST validate that these prefixes are within the scope of the DOTS client's domain. Other validation checks may be supported by DOTS servers. If this attribute is not provided, the DOTS server enforces the ACL on any destination IP address that belong to the DOTS client's domain.

This is an optional attribute.

source-ipv4-network: The source IPv4 prefix.

This is an optional attribute.

actions: Actions in the forwarding ACL category can be "drop" or "accept". The "accept" action is used to white-list traffic. The "drop" action is used to black-list traffic.

Accepted traffic may be subject to "rate-limit"; the allowed traffic rate is represented in bytes per second indicated in IEEE floating point format [IEEE.754.1985].

This is a mandatory attribute.

The DOTS server indicates the result of processing the POST request using the status-line header. Concretely, "201 Created" status-line MUST be returned in the response if the DOTS server has accepted the filtering rules. If the request is missing a mandatory attribute or contains an invalid or unknown parameter, "400 Bad Request" status-line MUST be returned by the DOTS server in the response. The error-tag is set to "missing-attribute", "invalid-value", or "unknown-element" as a function of the encountered error.

If the request is conflicting with an existing filtering installed by another DOTS client of the domain, the DOTS server returns "409

Conflict" status-line to the requesting DOTS client. The error-tag "resource-denied" is used in this case.

The "insert" query parameter (Section 4.8.5 of [RFC8040]) MAY be used to specify how an access control entry is inserted within an ACL and how an ACL is inserted within an ACL set.

The DOTS client MAY use the PUT request to modify its filtering rules maintained by the DOTS server.

8.2. Retrieve Installed Filtering Rules

The DOTS client periodically queries the DOTS server to check the counters for installed filtering rules. GET request is used to retrieve filtering rules from a DOTS server.

If the DOTS server does not find the access list name conveyed in the GET request in its configuration data for this DOTS client, it responds with a "404 Not Found" status-line.

Figure 15 shows how to retrieve all the filtering rules that were instantiated by the DOTS client and the number of matches for the installed filtering rules.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /access-lists?content=all HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 15: GET to Retrieve the Configuration Data and State Data for the Filtering Rules

Figure 16 shows how to retrieve "sample-ipv6-acl" filtering rule instantiated by the DOTS client, having "cuid=dz6pHjaADkaFTbjr0JGBpw", and the number of matches for the installed filtering rules.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw/access-lists\
    /acl=sample-ipv6-acl?content=all HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 16: GET to Retrieve the Configuration Data and State Data for a Filtering Rule

8.3. Remove Filtering Rules

DELETE request is used by a DOTS client to delete filtering rules from a DOTS server.

If the DOTS server does not find the access list name carried in the DELETE request in its configuration data for this DOTS client, it MUST respond with a "404 Not Found" status-line. The DOTS server successfully acknowledges a DOTS client's request to withdraw the filtering rules using "204 No Content" status-line, and removes the filtering rules accordingly.

Figure 17 shows an example of a request to remove the IPv4 ACL named "sample-ipv4-acl".

```
DELETE  /restconf/data/ietf-dots-data-channel:dots-data\
        /dots-client=dz6pHjaADkaFTbjr0JGBpw/access-lists\
        /acl=sample-ipv4-acl HTTP/1.1
Host: {host}:{port}
```

Figure 17: DELETE to Remove a Filtering Rule

9. IANA Considerations

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-dots-data-channel
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC7950].

```
name: ietf-dots-data-channel
namespace: urn:ietf:params:xml:ns:yang:ietf-dots-data-channel
prefix: data-channel
reference: RFC XXXX
```

10. Contributors

The following individuals have contributed to this document:

- Dan Wing, Email: dwing-ietf@fuggles.com

- Jon Shallow, NCC Group, Email: jon.shallow@nccgroup.trust

11. Security Considerations

RESTCONF security considerations are discussed in [RFC8040]. In particular, DOTS agents MUST follow the security recommendations in Sections 2 and 12 of [RFC8040]. Also, DOTS agents MUST support the mutual authentication TLS profile discussed in Sections 7.1 and 8 of [I-D.ietf-dots-signal-channel]. YANG ACL-specific security considerations are discussed in [I-D.ietf-netmod-acl-model].

Authenticated encryption MUST be used for data confidentiality and message integrity. The interaction between the DOTS agents requires Transport Layer Security (TLS) with a cipher suite offering confidentiality protection and the guidance given in [RFC7525] MUST be followed to avoid attacks on TLS.

An attacker may be able to inject RST packets, bogus application segments, etc., regardless of whether TLS authentication is used. Because the application data is TLS protected, this will not result in the application receiving bogus data, but it will constitute a DoS on the connection. This attack can be countered by using TCP-AO [RFC5925]. If TCP-AO is used, then any bogus packets injected by an attacker will be rejected by the TCP-AO integrity check and therefore will never reach the TLS layer.

In order to prevent leaking internal information outside a client-domain, client-side DOTS gateways SHOULD NOT reveal the identity of internal DOTS clients (e.g., source IP address, client's hostname) unless explicitly configured to do so.

DOTS servers MUST verify that requesting DOTS clients are entitled to enforce filtering rules on a given IP prefix. That is, only filtering rules on IP resources that belong to the DOTS client's domain MUST be authorized by a DOTS server.

Rate-limiting DOTS requests, including those with new 'cuid' values, from the same DOTS client defends against DoS attacks that would result in varying the 'cuid' to exhaust DOTS server resources. Rate-limit policies SHOULD be enforced on DOTS gateways (if deployed) and DOTS servers.

Applying resources quota per DOTS client and/or per DOTS client domain (e.g., limit the number of aliases and filters to be install by DOTS clients) prevents DOTS server resources to be aggressively used by some DOTS clients and ensures, therefore, DDoS mitigation usage fairness. Additionally, DOTS servers may limit the number of DOTS clients that can be enabled per domain.

All data nodes defined in the YANG module which can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations applied to these data nodes without proper protection can negatively affect network operations. Appropriate security measures are recommended to prevent illegitimate users from invoking DOTS data channel primitives. Nevertheless, an attacker who can access a DOTS client is technically capable of launching various attacks, such as:

- Set an arbitrarily low rate-limit, which may prevent legitimate traffic from being forwarded (rate-limit).

- Set an arbitrarily high rate-limit, which may lead to the forwarding of illegitimate DDoS traffic (rate-limit).

- Communicate invalid aliases to the server (alias), which will cause the failure of associating both data and signal channels.

- Set invalid ACL entries, which may prevent legitimate traffic from being forwarded. Likewise, invalid ACL entries may lead to forward DDoS traffic.

12. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Roman Danyliw, Ehud Doron, Russ White, Gilbert Clark, and Nesredien Suleiman for the discussion and comments.

13. References

13.1. Normative References

[I-D.ietf-dots-signal-channel]
Reddy, T., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-17 (work in progress), January 2018.

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Huang, L., Agarwal, S., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-15 (work in progress), January 2018.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[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>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

13.2. Informative References

[I-D.ietf-dots-architecture] Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-05 (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-12 (work in progress), January 2018.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-04 (work in progress), December 2017.

[IEEE.754.1985]
Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", August 1985.

[proto_numbers]
"IANA, "Protocol Numbers"", 2011, <http://www.iana.org/assignments/protocol-numbers>.

[RFC1983] Malkin, G., Ed., "Internet Users' Glossary", FYI 18, RFC 1983, DOI 10.17487/RFC1983, August 1996, <https://www.rfc-editor.org/info/rfc1983>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <https://www.rfc-editor.org/info/rfc6520>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

Authors' Addresses

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

Email: kondtir@gmail.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Email: kaname@nttv6.jp

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: frank.xialiang@huawei.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

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

Email: amortensen@arbor.net

Nik Teague
Verisign, Inc.
United States

Email: nteague@verisign.com