DOTS
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2017

T. Reddy
Cisco
M. Boucadair
Orange
K. Nishizuka
NTT Communications
L. Xia
Huawei
P. Patil
Cisco
A. Mortensen
Arbor Networks, Inc.
N. Teague
Verisign, Inc.

February 15, 2017

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

draft-reddy-dots-data-channel-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 19, 2017.

Table of Contents

1. Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale can be achieved by compromising enough end-hosts and using those infected hosts to perpetrate and

amplify the attack. The victim in this attack can be an application server, a client, a router, a firewall, or an entire network.

DDoS Open Threat Signaling (DOTS) defines two channels: signal and data channels [I-D.ietf-dots-architecture] (Figure 1). The DOTS signal channel used to convey that a network is under a DDOS attack to an upstream DOTS server so that appropriate mitigation actions are undertaken on the suspect traffic is further elaborated in [I-D.reddy-dots-signal-channel]. The DOTS data channel is used for infrequent bulk data exchange between DOTS agents in the aim to significantly augment attack response coordination.

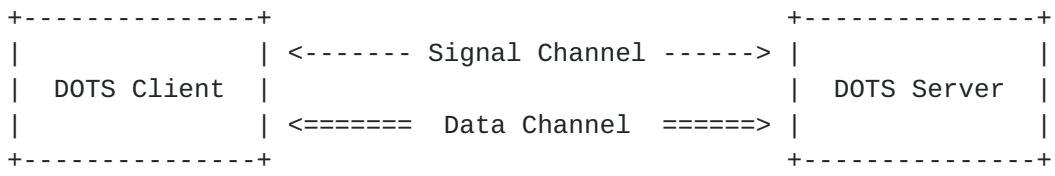

Figure 1: DOTS Channels

Section 2 of [I-D.ietf-dots-architecture] identifies that the DOTS data channel is used to perform the tasks listed below:

o Filter management, which enables a DOTS client to install or remove traffic filters, dropping or rate-limiting unwanted traffic and permitting white-listed traffic. Sample use cases for populating black- or white-list filtering rules are detailed hereafter:

   A. If a network resource (DOTS client) detects a potential DDoS attack from a set of IP addresses, the DOTS client informs its servicing router (DOTS gateway) of all suspect IP addresses that need to be blocked or black-listed for further investigation. The DOTS client could also specify a list of protocols and ports in the black-list rule. That DOTS gateway in-turn propagates the black-listed IP addresses to the DOTS server which will undertake appropriate action so that traffic from these IP addresses to the target network (specified by the DOTS client) is blocked.

   B. An enterprise network has partner sites from which only legitimate traffic arrives and the enterprise network wants to ensure that the traffic from these sites is not penalized during DDOS attacks. The DOTS client uses DOTS data channel to convey the white-listed IP addresses or prefixes of the partner sites to its DOTS server. The DOTS server uses this information to white-list flows from such IP addresses or prefixes reaching the enterprise network.

- Creating identifiers, such as names or aliases, for resources for which mitigation may be requested:

  A. The DOTS client may submit to the DOTS server a collection of prefixes it wants to refer to by alias when requesting mitigation, to which the server would respond with a success status and the new prefix group alias, or an error status and message in the event the DOTS client's data channel request failed (see requirement OP-006 in [I-D.ietf-dots-requirements] and Section 2 in [I-D.ietf-dots-architecture]).

2. Notational Conventions and Terminology

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

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

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

3. DOTS Data Channel

The DOTS data channel is intended to be used for bulk data exchanges between DOTS agents. Unlike the signal channel, which must operate nominally even when confronted with despite signal degradation due to packet loss, the data channel is not expected to be constructed to deal with attack conditions.

As the primary function of the data channel is data exchange, a reliable transport is required in order for DOTS agents to detect data delivery success or failure. RESTCONF [I-D.ietf-netconf-restconf] over TLS [RFC5246] over TCP is used for DOTS data channel (Figure 2). RESTCONF uses HTTP methods to provide CRUD operations on a conceptual datastore containing YANG-defined data, which is compatible with a server which implements NETCONF datastores. The HTTP POST, PUT, PATCH, and DELETE methods are used to edit data resources represented by DOTS data channel YANG data models. These basic edit operations allow the DOTS data channel running configuration to be altered by a DOTS client. DOTS data

channel configuration data and state data can be retrieved with the GET method. HTTP status codes are used to report success or failure for RESTCONF operations. The DOTS client will perform the root resource discovery procedure discussed in Section 3.1 of [I-D.ietf-netconf-restconf] to determine the root of the RESTCONF API. After discovering the RESTCONF API root, the DOTS client MUST use this value as the initial part of the path in the request URI, in any subsequent request to the DOTS server. The DOTS server can optionally support retrieval of the YANG modules it supports (Section 3.7 in [I-D.ietf-netconf-restconf]), for example, DOTS client can use RESTCONF to retreive the company proprietary YANG model supported by the DOTS server.

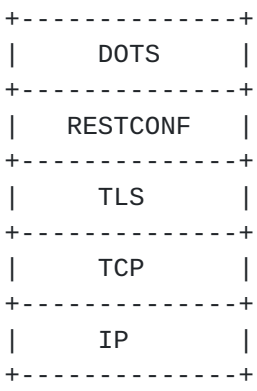

Figure 2: Abstract Layering of DOTS data channel over RESTCONF over TLS

JavaScript Object Notation (JSON) [RFC7159] payload is used to propogate data channel specific payload messages that convey request parameters and response information such as errors. This specification uses the encoding rules defined in [RFC7951] for representing DOTS data channel configuration data defined using YANG (Section 3.1) as JSON text.

A DOTS client registers itself to its DOTS server(s) in order to set up DOTS data channel related configuration data on the DOTS server and receive state data (i.e., non-configuration data) from the DOTS server. A single DOTS data channel between DOTS agents can be used to exchange multiple requests and multiple responses. To reduce DOTS client and DOTS server workload, DOTS client SHOULD re-use the 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 or NAT bindings. A TLS heartbeat [RFC6520] verifies the DOTS server still has TLS state by returning a TLS message.

3.1. DOTS Data Channel YANG Model

3.1.1. Identifier Model structure

This document defines a YANG [RFC6020] data model for creating identifers, such as names or aliases, for resources for which mitigation may be requested. Such identifiers may then be used in subsequent DOTS signal channel exchanges to refer more efficiently to the resources under attack.

This document defines the YANG module "ietf-dots-data-channel-identifier", which has the following structure:

```
module: ietf-dots-data-channel-identifier
    +--rw identifier
       +--rw alias* [alias-name]
          +--rw alias-name          string
          +--rw ip*                 inet:ip-address
          +--rw prefix*             inet:ip-prefix
          +--rw port-range* [lower-port upper-port]
          |  +--rw lower-port    inet:port-number
          |  +--rw upper-port    inet:port-number
          +--rw traffic-protocol*   uint8
          +--rw FQDN*               inet:domain-name
          +--rw URI*                inet:uri
          +--rw E.164*              string
```

3.1.2. Identifier Model

```
<CODE BEGINS> file "ietf-dots-data-channel-identifier@2016-11-28.yang"

module ietf-dots-data-channel-identifier {
      namespace "urn:ietf:params:xml:ns:yang:ietf-dots-data-channel-
identifier";
      prefix "alias";
      import ietf-inet-types {
          prefix "inet";
      }
     organization "Cisco Systems, Inc.";
     contact "Tirumaleswar Reddy <tireddy@cisco.com>";

     description
       "This module contains YANG definition for
        configuring identifiers for resources using DOTS data channel";

     revision 2016-11-28 {
       reference
       "https://tools.ietf.org/html/draft-reddy-dots-data-channel";
     }
```

```
     container identifier {
          description "top level container for identifiers";
              list alias {
                   key alias-name;
                   description "list of identifiers";
                   leaf alias-name {
                      type string;
                      description "alias name";
                   }
                   leaf-list ip {
                      type inet:ip-address;
                      description "IP address";
                   }
                   leaf-list prefix {
                      type inet:ip-prefix;
                      description "prefix";
                   }
                   list 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";
                      }
                      leaf upper-port {
                         type inet:port-number;
                         must ". >= ../lower-port" {
                           error-message
                           "The upper-port must be greater than or
                            equal to lower-port";
                         }
                         description "upper port";
                      }
                   }
                   leaf-list traffic-protocol {
                      type uint8;
                      description "Internet Protocol number";
                   }
                   leaf-list FQDN {
                     type inet:domain-name;
                     description "FQDN";
                   }
                   leaf-list URI {
                     type inet:uri;
                     description "URI";
                   }
```

```
                  leaf-list E.164 {
                     type string;
                     description "E.164 number";
                  }
               }
     }
  }
 <CODE ENDS>
```

3.1.3. Filter Model and structure

This document uses the Access Control List (ACL) YANG data model [I-D.ietf-netmod-acl-model] for the configuration of filtering rules. ACL is explained in Section 1 of [I-D.ietf-netmod-acl-model].

Examples of such configuration include:

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

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

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

3.2. Identifiers

3.2.1. Create Identifiers

A POST request is used to create identifiers, such as names or aliases, for resources for which a mitigation may be requested. Such identifiers may then be used in subsequent DOTS signal channel exchanges to refer more efficiently to the resources under attack (Figure 3).

```
 POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
 Host: {host}:{port}
 Content-Format: "application/yang.api+json"
 {
  "ietf-dots-data-channel-identifier:identifier": {
    "alias": [
      {
        "alias-name": "string",
        "ip": [
          "string"
        ],
        "prefix": [
          "string"
        ],
        "port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
        ],
        "traffic-protocol": [
          integer
        ],
        "FQDN": [
          "string"
        ],
        "URI": [
          "string"
        ],
        "E.164": [
          "string"
        ]
      }
    ]
  }
}
```

Figure 3: POST to create identifiers

The header parameters are described below:

alias-name: Name of the alias. This is a mandatory attribute.

traffic-protocol: Internet Protocol numbers. This is an optional attribute.

port-range: The port range, lower-port for lower port number and upper-port for upper port number. For TCP, UDP, SCTP, or DCCP: the range of ports (e.g., 80 to 8080). This is an optional attribute.

ip: IP addresses are separated by commas. This is an optional attribute.

prefix: Prefixes are separated by commas. This is an optional attribute.

FQDN: Fully Qualified Domain Name, is the full name of a system, rather than just its hostname. For example, "venera" is a hostname, and "venera.isi.edu" is an FQDN. This is an optional attribute.

URI: Uniform Resource Identifier (URI). This is an optional attribute.

E.164: E.164 number. This is an optional attribute.

In the POST request at least one of the attributes ip or prefix or FQDN or URI MUST be present. DOTS agents can safely ignore Vendor-Specific parameters they don't understand.

Figure 4 shows a POST request to create alias called "https1" for HTTP(S) servers with IP addresses 2002:db8:6401::1 and 2002:db8:6401::2 listening on port 443.

```
POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
  "ietf-dots-data-channel-identifier:identifier": {
    "alias": [
      {
        "alias-name": "Server1",
        "traffic-protocol": [
          6
        ],
        "ip": [
          "2002:db8:6401::1",
          "2002:db8:6401::2"
        ],
        "port-range": [
          {
            "lower-port": 443
          }
        ]
      }
    ]
  }
}
```

Figure 4: POST to create identifiers

The DOTS server indicates the result of processing the POST request using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx codes are some sort of invalid requests and 5xx codes are returned if the DOTS server has erred or it is incapable of accepting the alias. Response code 201 (Created) will be returned in the response if the DOTS server has accepted the alias. If the request is missing one or more mandatory attributes then 400 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 400 (Invalid query) will be returned in the response. The HTTP response will include the JSON body received in the request.

The DOTS client can use the PUT request (Section 4.5 in [I-D.ietf-netmod-acl-model]) to create or modify the aliases in the DOTS server.

3.2.2. Delete Identifiers

A DELETE request is used to delete identifiers maintained by a DOTS server (Figure 5).

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

Figure 5: DELETE identifier

In RESTCONF, URI-encoded path expressions are used. A RESTCONF data resource identifier is encoded from left to right, starting with the top-level data node, according to the "api-path" rule defined in Section 3.5.3.1 of [I-D.ietf-netconf-restconf]. The data node in the above path expression is a YANG list node and MUST be encoded according to the rules defined in Section 3.5.1 of [I-D.ietf-netconf-restconf].

If the DOTS server does not find the alias name conveyed in the DELETE request in its configuration data, then it responds with a 404 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client's request to remove the identifier using 204 (No Content) in the response.

3.2.3. Retrieving Installed Identifiers

A GET request is used to retrieve the set of installed identifiers from a DOTS server (Section 3.3.1 in [I-D.ietf-netconf-restconf]). Figure 6 shows how to retrieve all the identifiers that were instantiated by the DOTS client. The content parameter and its permitted values are defined in Section 4.8.1 of [I-D.ietf-netconf-restconf].

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

Figure 6: GET to retrieve all the installed identifiers

Figure 7 shows response for all identifiers on the DOTS server.

```
{
 "ietf-dots-data-channel-identifier:identifier": [
  {
    "alias": [
      {
        "alias-name": "Server1",
        "traffic-protocol": [
          6
        ],
        "ip": [
          "2002:db8:6401::1",
          "2002:db8:6401::2"
        ],
        "port-range": [
          {
            "lower-port": 443
          }
        ]
      }
    ]
  },
  {
    "alias": [
      {
        "alias-name": "Server2",
        "traffic-protocol": [
          6
        ],
        "ip": [
          "2002:db8:6401::10",
          "2002:db8:6401::20"
        ],
        "port-range": [
          {
            "lower-port": 80
          }
        ]
      }
    ]
  }
 ]
}
```

Figure 7: Response body

If the DOTS server does not find the alias name conveyed in the GET request in its configuration data, then it responds with a 404 (Not Found) error response code.

3.3. Filtering Rules

The DOTS server either receives the filtering rules directly from the DOTS client or via the DOTS gateway. If the DOTS client signals the filtering rules via the DOTS gateway then the DOTS gateway validates if the DOTS client is authorized to signal the filtering rules and if the client is authorized propagates the rules to the DOTS server. Likewise, the DOTS server validates if the DOTS gateway is authorized to signal the filtering rules. To create or purge filters, the DOTS client sends HTTP requests to the DOTS gateway. The DOTS gateway validates the rules in the requests and proxies the requests containing the filtering rules to a DOTS server. When the DOTS gateway receives the associated HTTP response from the DOTS server, it propagates the response back to the DOTS client.

The following APIs define means for a DOTS client to configure filtering rules on a DOTS server.

3.3.1. Install Filtering Rules

A POST request is used to push filtering rules to a DOTS server. Figure 8 shows a POST request example to block traffic from 10.10.10.1/24, destined to 11.11.11.1/24. The ACL JSON configuration for the filtering rule is generated using the ACL YANG data model defined in [I-D.ietf-netmod-acl-model] and the ACL configuration XML for the filtering rule is specified in Section 4.3 of [I-D.ietf-netmod-acl-model]. This specification updates the ACL YANG data model defined in [I-D.ietf-netmod-acl-model] to support rate-limit action.

```
POST /restconf/data/ietf-access-control-list HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
 "ietf-access-control-list:access-lists": {
    "acl": [
        {
             "acl-name": "sample-ipv4-acl",
             "acl-type": "ipv4",
             "access-list-entries": {
                 "ace": [
                     {
                         "rule-name": "rule1",
                         "matches": {
                             "source-ipv4-network": "10.10.10.1/24",
                             "destination-ipv4-network": "11.11.11.1/24"
                          },
                          "actions": {
                              "deny": [null]
                          }
                     }
                 ]
             }
        }
    ]
 }
}
```

Figure 8: POST to install filterng rules

The header parameters defined in [I-D.ietf-netmod-acl-model] are discussed below:

acl-name: The name of access-list. This is a mandatory attribute.

acl-type: Indicates the primary intended type of match criteria (e.g. IPv4, IPv6). This is a mandatory attribute.

protocol: Internet Protocol numbers. This is an optional attribute.

source-ipv4-network: The source IPv4 prefix. This is an optional attribute.

destination-ipv4-network: The destination IPv4 prefix. This is an optional attribute.

actions: "deny" or "permit" or "rate-limit". "permit" action is used to white-list traffic. "deny" action is used to black-list traffic. "rate-limit" action is used to rate-limit traffic, the allowed traffic rate is represented in bytes per second indicated in IEEE floating point format [IEEE.754.1985]. If actions attribute is not specified in the request then the default action is "deny". This is an optional attribute.

The DOTS server indicates the result of processing the POST request using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx codes are some sort of invalid requests and 5xx codes are returned if the DOTS server has erred or it is incapable of configuring the filtering rules. Response code 201 (Created) will be returned in the response if the DOTS server has accepted the filtering rules. If the request is missing one or more mandatory attributes then 400 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 400 (Invalid query) will be returned in the response.

The DOTS client can use the PUT request to create or modify the filtering rules in the DOTS server.

3.3.2. Remove Filtering Rules

A DELETE request is used to delete filtering rules from a DOTS server (Figure 9).

```
DELETE /restconf/data/ietf-access-control-list:access-lists/acl-name\
       =sample-ipv4-acl&acl-type=ipv4 HTTP/1.1
Host: {host}:{port}
```

Figure 9: DELETE to remove the filtering rules

If the DOTS server does not find the access list name and access list type conveyed in the DELETE request in its configuration data, then it responds with a 404 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client's request to withdraw the filtering rules using 204 (No Content) response code, and removes the filtering rules as soon as possible.

3.3.3. Retrieving Installed Filtering Rules

The DOTS client periodically queries the DOTS server to check the counters for installed filtering rules. A GET request is used to retrieve filtering rules from a DOTS server. Figure 10 shows how to retrieve all the filtering rules programmed by the DOTS client and the number of matches for the installed filtering rules.

```
GET /restconf/data/ietf-access-control-list:access-lists?content=all HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 10: GET to retrieve the configuration data and state data for the filtering rules

If the DOTS server does not find the access list name and access list type conveyed in the GET request in its configuration data, then it responds with a 404 (Not Found) error response code.

4. IANA Considerations

TODO

[TBD: DOTS WG will have to do something similar to https://tools.ietf.org/html/rfc7519#section-10, and create JSON DOTS claim registry and register the JSON attributes defined in this specification].

5. Contributors

The following individuals have contributed to this document:

Dan Wing Email: dwing-ietf@fuggles.com

6. Security Considerations

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

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

Special care should be taken in order to ensure that the activation of the proposed mechanism won't have an impact on the stability of the network (including connectivity and services delivered over that network).

Involved functional elements in the cooperation system must establish exchange instructions and notification over a secure and authenticated channel. Adequate filters can be enforced to avoid that nodes outside a trusted domain can inject request such as deleting filtering rules. Nevertheless, attacks can be initiated from within the trusted domain if an entity has been corrupted. Adequate means to monitor trusted nodes should also be enabled.

7. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Andrew Mortensen, Roman Danyliw, Ehud Doron and Gilbert Clark for the discussion and comments.

8. References

8.1. Normative References

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

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

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

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

[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, <http://www.rfc-editor.org/info/rfc7525>.

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

8.2. Informative References

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-03 (work in progress), October 2016.

[I-D.ietf-netmod-acl-model] Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-09 (work in progress), October 2016.

[I-D.reddy-dots-signal-channel] Reddy, T., Boucadair, M., and P. Patil, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", draft-reddy-dots-signal-channel-07 (work in progress), December 2016.

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

[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, <http://www.rfc-editor.org/info/rfc4632>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[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, <http://www.rfc-editor.org/info/rfc6520>.

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

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

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

Email: kaname@nttv6.jp

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

Email: frank.xialiang@huawei.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

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

Email: amortensen@arbor.net

Nik Teague
Verisign, Inc.
United States

Email: nteague@verisign.com