Internet Engineering Task Force (IETF)
Request for Comments: 9133
Category: Standards Track
ISSN: 2070-1721

K. Nishizuka
NTT Communications
M. Boucadair
Orange
T. Reddy.K
Akamai
T. Nagata
Lepidum

September 2021

Controlling Filtering Rules Using Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel

Abstract

This document specifies an extension to the Distributed Denial-of-Service Open Threat Signaling (DOTS) signal channel protocol so that DOTS clients can control their filtering rules when an attack mitigation is active.

Particularly, this extension allows a DOTS client to activate or deactivate existing filtering rules during a Distributed Denial-of-Service (DDoS) attack. The characterization of these filtering rules is conveyed by a DOTS client during an 'idle' time (i.e., no mitigation is active) by means of the DOTS data channel protocol.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9133.

Table of Contents

1. Introduction

1.1. The Problem

In the Distributed Denial-of-Service Open Threat Signaling (DOTS) architecture [RFC8811], DOTS clients and servers communicate using both a signal channel protocol [RFC9132] and a data channel protocol [RFC8783].

The DOTS data channel protocol [RFC8783] is used for bulk data exchange between DOTS agents to improve the coordination of parties involved in the response to a Distributed Denial-of-Service (DDoS) attack. Filter management, which is one of the tasks of the DOTS data channel protocol, enables a DOTS client to retrieve the filtering capabilities of a DOTS server and to manage filtering rules. Typically, these filtering rules are used for dropping or rate-limiting unwanted traffic, and permitting accept-listed traffic.

The DOTS signal channel protocol [RFC9132] is designed to be resilient under extremely hostile network conditions and provides continued contact between DOTS agents even as DDoS attack traffic saturates the link. The DOTS signal channel can be established between two DOTS agents prior to or during an attack. At any time, the DOTS client may send mitigation requests (as per Section 4.4 of [RFC9132]) to a DOTS server over the active signal channel. While mitigation is active, the DOTS server periodically sends status messages to the DOTS client, including basic mitigation feedback details. In case of a massive DDoS attack that saturates the incoming link(s) to the DOTS client, all traffic from the DOTS server

to the DOTS client will likely be dropped. However, the DOTS server may still receive DOTS messages sent from the DOTS client over the signaling channel thanks to the heartbeat requests keeping the channel active (as described in Section 4.7 of [RFC9132]).

Unlike the DOTS signal channel protocol, the DOTS data channel protocol is not expected to deal with attack conditions. As such, an issue that might be encountered in some deployments is when filters installed by means of the DOTS data channel protocol may not function as expected during DDoS attacks or, worse, exacerbate an ongoing DDoS attack. In such conditions, the DOTS data channel protocol cannot be used to change these filters, which may complicate DDoS mitigation operations [INTEROP].

A typical case is a conflict between filtering rules installed by a DOTS client and the mitigation actions of a DDoS mitigator. Consider, for instance, a DOTS client that configures during 'idle' time (i.e., no mitigation is active) some filtering rules using the DOTS data channel protocol to permit traffic from accept-listed sources. However, during a volumetric DDoS attack, the DDoS mitigator identifies the source addresses/prefixes in the accept-listed filtering rules are attacking the target. For example, an attacker can spoof the IP addresses of accept-listed sources to generate attack traffic, or the attacker can compromise the accept-listed sources and program them to launch a DDoS attack.

[RFC9132] is designed so that the DDoS server notifies the above conflict to the DOTS client (that is, the 'conflict-cause' parameter is set to 2 (conflict-with-acceptlist)), but the DOTS client may not be able to withdraw the accept-list rules during the attack period due to the high-volume attack traffic saturating the inbound link to the DOTS client domain. In other words, the DOTS client cannot use the DOTS data channel protocol to withdraw the accept-list filters when a DDoS attack is in progress.

1.2. Controlling Filtering Rules Using DOTS Signal Channel

This specification addresses the problems discussed in Section 1.1 by adding a capability for managing filtering rules using the DOTS signal channel protocol, which enables a DOTS client to request the activation (or deactivation) of filtering rules during a DDoS attack. Note that creating these filtering rules is still the responsibility of the DOTS data channel [RFC8783].

The DOTS signal channel protocol is designed to enable a DOTS client to contact a DOTS server for help even under severe network congestion conditions. Therefore, extending the DOTS signal channel protocol to manage the filtering rules during an attack will enhance the protection capability offered by DOTS protocols.

> Note: The experiment at the IETF 103 hackathon [INTEROP] showed that even when the inbound link is saturated by DDoS attack traffic, the DOTS client can signal mitigation requests using the DOTS signal channel over the saturated link.

Conflicts that are induced by filters installed by other DOTS clients

of the same domain are not discussed in this specification.

An augmentation to the DOTS signal channel YANG module is defined in Section 3.2.2.

Sample examples are provided in Section 4, in particular:

* Section 4.1 illustrates how the filter control extension is used when conflicts with Access Control Lists (ACLs) are detected and reported by a DOTS server.

* Section 4.2 shows how a DOTS client can instruct a DOTS server to safely forward some specific traffic in 'attack' time.

* Section 4.3 shows how a DOTS client can react if the DDoS traffic is still being forwarded to the DOTS client domain even if mitigation requests were sent to a DOTS server.

The JavaScript Object Notation (JSON) encoding of YANG-modeled data [RFC7951] is used to illustrate the examples.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The reader should be familiar with the terms defined in [RFC8612].

The terminology for describing YANG modules is defined in [RFC7950]. The meaning of the symbols in the tree diagram is defined in [RFC8340] and [RFC8791].

3. Controlling Filtering Rules of a DOTS Client

3.1. Binding DOTS Data and Signal Channels

The filtering rules eventually managed using the DOTS signal channel protocol are created a priori by the same DOTS client using the DOTS data channel protocol. Managing conflicts with filters installed by other DOTS clients of the same domain is out of scope.

As discussed in Section 4.4.1 of [RFC9132], a DOTS client must use the same 'cuid' for both the DOTS signal and data channels. This requirement is meant to facilitate binding DOTS channels used by the same DOTS client.

The DOTS signal and data channels from a DOTS client may or may not use the same DOTS server. Nevertheless, the scope of the mitigation request, alias, and filtering rules are not restricted to the DOTS server but to the DOTS server domain. To that aim, DOTS servers within a domain are assumed to have a mechanism to coordinate the mitigation requests, aliases, and filtering rules to coordinate their decisions for better mitigation operation efficiency. The exact

details about such a mechanism is out of the scope of this document.

A filtering rule controlled by the DOTS signal channel is identified by its ACL name (Section 4.3 of [RFC8783]). Note that an ACL name unambiguously identifies an ACL bound to a DOTS client, but the same name may be used by distinct DOTS clients.

The activation or deactivation of an ACL by the DOTS signal channel overrides the 'activation-type' (defined in Section 4.3 of [RFC8783]) a priori conveyed with the filtering rules using the DOTS data channel protocol.

Once the attack is mitigated, the DOTS client may use the data channel to control the 'activation-type' (e.g., revert to a default value) of some of the filtering rules controlled by the DOTS signal channel or delete some of these filters. This behavior is deployment specific.

3.2. DOTS Signal Channel Extension

3.2.1. Parameters and Behaviors

This specification extends the mitigation request defined in Section 4.4.1 of [RFC9132] to convey the intended control of configured filtering rules. Concretely, the DOTS client conveys the 'acl-list' attribute with the following sub-attributes in the Concise Binary Object Representation (CBOR) body of a mitigation request (see the YANG structure in Section 3.2.2.1):

acl-name: A name of an access list defined using the DOTS data channel (Section 4.3 of [RFC8783]) that is associated with the DOTS client.

As a reminder, an ACL is an ordered list of Access Control Entries (ACEs). Each ACE has a list of match criteria and a list of actions [RFC8783]. The list of configured ACLs can be retrieved using the DOTS data channel during 'idle' time.

This is a mandatory attribute when 'acl-list' is included.

activation-type: An attribute indicating the activation type of an ACL overriding the existing 'activation-type' installed by the DOTS client using the DOTS data channel.

As a reminder, this attribute can be set to 'deactivate', 'immediate', or 'activate-when-mitigating' as defined in [RFC8783].

Note that both 'immediate' and 'activate-when-mitigating' have an immediate effect when a mitigation request is being processed by the DOTS server.

This is an optional attribute.

By default, ACL-related operations are achieved using the DOTS data channel protocol when no attack is ongoing. DOTS clients MUST NOT

use the filtering control over the DOTS signal channel in 'idle' time; such requests MUST be discarded by DOTS servers with 4.00 (Bad Request).

During an attack time, DOTS clients may include 'acl-list', 'acl-name', and 'activation-type' attributes in a mitigation request. This request may be the initial mitigation request for a given mitigation scope or a new one overriding an existing request. In both cases, a new 'mid' MUST be used. Nevertheless, it is NOT RECOMMENDED to include ACL attributes in an initial mitigation request for a given mitigation scope or in a mitigation request adjusting the mitigation scope. This recommendation is meant to avoid delaying attack mitigations because of failures to process ACL attributes.

As the attack evolves, DOTS clients can adjust the 'activation-type' of an ACL conveyed in a mitigation request or control other filters as necessary. This can be achieved by sending a PUT request with a new 'mid' value.

It is RECOMMENDED for a DOTS client to subscribe to asynchronous notifications of the attack mitigation, as detailed in Section 4.4.2.1 of [RFC9132]. If not, the polling mechanism in Section 4.4.2.2 of [RFC9132] has to be followed by the DOTS client.

A DOTS client relies on the information received from the DOTS server and/or local information to the DOTS client domain to trigger a filter control request. Only filters that are pertinent for an ongoing mitigation should be controlled by a DOTS client using the DOTS signal channel.

'acl-list', 'acl-name', and 'activation-type' are defined as comprehension-required parameters. Following the rules in Section 6 of [RFC9132], if the DOTS server does not understand the 'acl-list', 'acl-name', or 'activation-type' attributes, it responds with a 4.00 (Bad Request) error response code.

If the DOTS server does not find the ACL name ('acl-name') conveyed in the mitigation request for this DOTS client, it MUST respond with a 4.04 (Not Found) error response code.

If the DOTS server finds the ACL name for this DOTS client, and assuming the request passed the validation checks in Section 4.4.1 of [RFC9132], the DOTS server MUST proceed with the 'activation-type' update. The update is immediately enforced by the DOTS server and will be maintained as the new activation type for the ACL name even after the termination of the mitigation request. In addition, the DOTS server MUST update the lifetime of the corresponding ACL similar to the update when a refresh request is received using the DOTS data channel (Section 7.2 of [RFC8783]). If, for some reason, the DOTS server fails to apply the filter update, it MUST respond with a 5.03 (Service Unavailable) error response code and include the failed ACL update in the diagnostic payload of the response (an example is shown in Figure 1). Else, the DOTS server replies with the appropriate response code defined in Section 4.4.1 of [RFC9132].

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 123,
        "ietf-dots-signal-control:acl-list": [
          {
            "acl-name": "an-accept-list",
            "activation-type": "deactivate"
          }
        ]
      }
    ]
  }
}
```

Figure 1: Example of a Diagnostic Payload Including Failed ACL Update

The JSON/YANG mappings for DOTS filter control attributes are shown in Table 1. As a reminder, the mapping for 'acl-name' is defined in Table 5 of [RFC9132].

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
activation-type	enumeration	52	0 unsigned	String
ietf-dots-signal-control:acl-list	list	53	4 array	Array
acl-name	leafref	23	3 text string	String

Table 1: JSON/YANG Mapping to CBOR for Filter Control Attributes

If the DOTS client receives a 5.03 (Service Unavailable) with a diagnostic payload indicating a failed ACL update as a response to an initial mitigation or a mitigation with adjusted scope, the DOTS client MUST immediately send a new request that repeats all the parameters as sent in the failed mitigation request but without including the ACL attributes. After the expiry of Max-Age returned in the 5.03 (Service Unavailable) response, the DOTS client retries with a new mitigation request (i.e., a new 'mid') that repeats all the parameters as sent in the failed mitigation request (i.e., the one including the ACL attributes).

If, during an active mitigation, the 'activation-type' is changed at the DOTS server (e.g., as a result of an external action) for an ACL bound to a DOTS client, the DOTS server notifies that DOTS client of the change by including the corresponding ACL parameters in an asynchronous notification (the DOTS client is observing the active mitigation) or in a response to a polling request (Section 4.4.2.2 of [RFC9132]).

If the DOTS signal and data channels of a DOTS client are not established with the same DOTS server of a DOTS server domain, the above request processing operations are undertaken using the coordination mechanism discussed in Section 3.1.

This specification does not require any modification to the efficacy update and the withdrawal procedures defined in [RFC9132]. In particular, ACL-related clauses are not included in a PUT request used to send an efficacy update and DELETE requests.

3.2.2. DOTS Signal Filtering Control Module

3.2.2.1. Tree Structure

This document augments the "ietf-dots-signal-channel" YANG module defined in [RFC9132] for managing filtering rules.

This document defines the YANG module "ietf-dots-signal-control", which has the following tree structure:

```
module: ietf-dots-signal-control
 augment-structure /dots-signal:dots-signal/dots-signal:message-type
                   /dots-signal:mitigation-scope/dots-signal:scope:
   +-- acl-list* [acl-name]
      +-- acl-name
      |       -> /data-channel:dots-data/dots-client/acls/acl/name
      +-- activation-type?   data-channel:activation-type
```

3.2.2.2. YANG Module

This YANG module is not intended to be used via NETCONF/RESTCONF for DOTS server management purposes; such a module is out of the scope of this document. It serves only to provide a data model and encoding, but not a management data model.

This module uses types defined in [RFC8783].

```
<CODE BEGINS> file "ietf-dots-signal-control@2021-09-02.yang"
module ietf-dots-signal-control {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal-control";
  prefix dots-control;

  import ietf-dots-signal-channel {
    prefix dots-signal;
    reference
      "RFC 9132: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Signal Channel Specification";
  }


  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "RFC 8791: YANG Data Structure Extensions";
  }
```

```
import ietf-dots-data-channel {
  prefix data-channel;
  reference
    "RFC 8783: Distributed Denial-of-Service Open Threat
                Signaling (DOTS) Data Channel Specification";
}

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

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

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

   Author:  Tirumaleswar Reddy.K
            <mailto:kondtir@gmail.com>

   Author:  Takahiko Nagata
            <mailto:nagata@lepidum.co.jp>";

description
  "This module contains YANG definition for the signaling
   messages exchanged between a DOTS client and a DOTS server
   to control, by means of the DOTS signal channel, filtering
   rules configured using the DOTS data channel.

   Copyright (c) 2021 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
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC 9133; see
   the RFC itself for full legal notices.";

revision 2021-09-02 {
  description
    "Initial revision.";
  reference
    "RFC 9133: Controlling Filtering Rules Using Distributed
                Denial-of-Service Open Threat Signaling (DOTS)
                Signal Channel";
}

sx:augment-structure "/dots-signal:dots-signal"
                   + "/dots-signal:message-type"
```

```
                    + "/dots-signal:mitigation-scope"
                    + "/dots-signal:scope" {

      description
        "ACL name and activation type.";

      list acl-list {
        key "acl-name";
        description
          "List of ACLs as defined using the DOTS data
           channel. ACLs bound to a DOTS client are uniquely
           identified by a name.";
        leaf acl-name {
          type leafref {
            path "/data-channel:dots-data/data-channel:dots-client"
               + "/data-channel:acls/data-channel:acl"
               + "/data-channel:name";
          }
          description
            "Reference to the ACL name bound to a DOTS client.";
        }
        leaf activation-type {
          type data-channel:activation-type;
          default "activate-when-mitigating";
          description
            "Sets the activation type of an ACL.";
        }
      }
    }
  }
  <CODE ENDS>
```

4. Some Examples

This section provides some examples to illustrate the behavior specified in Section 3.2.1. These examples are provided for illustration purposes; they should not be considered as deployment recommendations.

4.1. Conflict Handling

Let's consider a DOTS client that contacts its DOTS server during 'idle' time to install an accept-list allowing for UDP traffic issued from 2001:db8:1234::/48 with a destination port number 443 to be forwarded to 2001:db8:6401::2/127. It does so by sending, for example, a PUT request as shown in Figure 2.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
    /acl=an-accept-list HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [
```

```
      {
        "name": "an-accept-list",
        "type": "ipv6-acl-type",
        "activation-type": "activate-when-mitigating",
        "aces": {
          "ace": [
            {
              "name": "test-ace-ipv6-udp",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8:6401::2/127",
                  "source-ipv6-network": "2001:db8:1234::/48"
                },
                "udp": {
                  "destination-port-range-or-operator": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 2: DOTS Data Channel Request to Create a Filter

Sometime later, consider that a DDoS attack is detected by the DOTS client on 2001:db8:6401::2/127. Consequently, the DOTS client sends a mitigation request to its DOTS server as shown in Figure 3.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=paL8p4Zqo4SLv64TLPXrxA"
Uri-Path: "mid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:6401::2/127"
        ],
        "target-protocol": [
          17
        ],
        "lifetime": 3600
```

```
      }
    ]
  }
}
```

Figure 3: DOTS Signal Channel Mitigation Request

The DOTS server immediately accepts the request by replying with 2.01 (Created) (Figure 4 depicts the message body of the response).

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 123,
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 4: Status Response (Message Body)

Assuming the DOTS client subscribed to asynchronous notifications, when the DOTS server concludes that some of the attack sources belong to 2001:db8:1234::/48, it sends a notification message with 'status' code set to 1 (attack-mitigation-in-progress) and 'conflict-cause' set to 2 (conflict-with-acceptlist) to the DOTS client to indicate that this mitigation request is in progress, but a conflict is detected.

Upon receipt of the notification message from the DOTS server, the DOTS client sends a PUT request to deactivate the "an-accept-list" ACL as shown in Figure 5.

The DOTS client can also decide to send a PUT request to deactivate the "an-accept-list" ACL if suspect traffic is received from an accept-listed source (2001:db8:1234::/48). The structure of that PUT is the same as the one shown in Figure 5.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=paL8p4Zqo4SLv64TLPXrxA"
Uri-Path: "mid=124"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:6401::2/127"
        ],
        "target-protocol": [
```

```
          17
        ],
        "ietf-dots-signal-control:acl-list": [
          {
            "acl-name": "an-accept-list",
            "activation-type": "deactivate"
          }
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 5: PUT for Deactivating a Conflicting Filter

Then, the DOTS server deactivates the "an-accept-list" ACL and replies with a 2.04 (Changed) response to the DOTS client to confirm the successful operation. The message body is similar to the one depicted in Figure 4.

Once the attack is mitigated, the DOTS client may use the data channel to retrieve its ACLs maintained by the DOTS server. As shown in Figure 6, the activation type is set to 'deactivate' as set by the DOTS signal channel (Figure 5) instead of the type initially set using the DOTS data channel (Figure 2).

```
{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "an-accept-list",
        "type": "ipv6-acl-type",
        "activation-type": "deactivate",
        "pending-lifetime": 10021,
        "aces": {
          "ace": [
            {
              "name": "test-ace-ipv6-udp",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8:6401::2/127",
                  "source-ipv6-network": "2001:db8:1234::/48"
                },
                "udp": {
                  "destination-port-range-or-operator": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
```

```
          }
        }
      ]
    }
  }
```

Figure 6: DOTS Data Channel GET Response after Mitigation (Message Body)

4.2. On-Demand Activation of an Accept-List Filter

Let's consider a DOTS client that contacts its DOTS server during 'idle' time to install an accept-list allowing for UDP traffic issued from 2001:db8:1234::/48 to be forwarded to 2001:db8:6401::2/127. It does so by sending, for example, a PUT request shown in Figure 7. The DOTS server installs this filter with a "deactivated" state.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=ioiuLoZqo4SLv64TLPXrxA/acls\
    /acl=my-accept-list HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "my-accept-list",
        "type": "ipv6-acl-type",
        "activation-type": "deactivate",
        "aces": {
          "ace": [
            {
              "name": "an-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8:6401::2/127",
                  "source-ipv6-network": "2001:db8:1234::/48",
                  "protocol": 17
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 7: DOTS Data Channel Request to Create an Accept-List Filter

Sometime later, consider that a UDP DDoS attack is detected by the DOTS client on 2001:db8:6401::2/127 but the DOTS client wants to let

the traffic from 2001:db8:1234::/48 be accept-listed to the DOTS client domain. Consequently, the DOTS client sends a mitigation request to its DOTS server as shown in Figure 8.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=ioiuLoZqo4SLv64TLPXrxA"
Uri-Path: "mid=4879"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:6401::2/127"
        ],
        "target-protocol": [
          17
        ],
        "ietf-dots-signal-control:acl-list": [
          {
            "acl-name": "my-accept-list",
            "activation-type": "immediate"
          }
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 8: DOTS Signal Channel Mitigation Request with a Filter Control

The DOTS server activates the "my-accept-list" ACL and replies with a 2.01 (Created) response to the DOTS client to confirm the successful operation.

4.3. DOTS Servers/Mitigators Lacking Capacity

This section describes a scenario in which a DOTS client activates a drop-list or a rate-limit filter during an attack.

Consider a DOTS client that contacts its DOTS server during 'idle' time to install an accept-list that rate-limits all (or a part thereof) traffic to be forwarded to 2001:db8:123::/48 as a last resort countermeasure whenever required. Installing the accept-list can be done by sending, for example, the PUT request shown in Figure 9. The DOTS server installs this filter with a "deactivated" state.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=OopPisZqo4SLv64TLPXrxA/acls\
```

```
    /acl=my-ratelimit-list HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "my-ratelimit-list",
        "type": "ipv6-acl-type",
        "activation-type": "deactivate",
        "aces": {
          "ace": [
            {
              "name": "my-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network": "2001:db8:123::/48"
                }
              },
              "actions": {
                "forwarding": "accept",
                "rate-limit": "20000.00"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 9: DOTS Data Channel Request to Create a Rate-Limit Filter

Consider now that a DDoS attack is detected by the DOTS client on 2001:db8:123::/48. Consequently, the DOTS client sends a mitigation request to its DOTS server (Figure 10).

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=OopPisZqo4SLv64TLPXrxA"
Uri-Path: "mid=85"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:123::/48"
        ],
        "lifetime": 3600
      }
    ]
```

```
  }
}
```

Figure 10: DOTS Signal Channel Mitigation Request

For some reason (e.g., the DOTS server, or the mitigator, is lacking a capability or capacity), the DOTS client is still receiving attack traffic, which saturates available links. To soften the problem, the DOTS client decides to activate the filter that rate-limits the traffic destined to the DOTS client domain. To that aim, the DOTS client sends the mitigation request to its DOTS server shown in Figure 11.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=OopPisZqo4SLv64TLPXrxA"
Uri-Path: "mid=86"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:123::/48"
        ],
        "ietf-dots-signal-control:acl-list": [
          {
            "acl-name": "my-ratelimit-list",
            "activation-type": "immediate"
          }
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 11: DOTS Signal Channel Mitigation Request to Activate a Rate-Limit Filter

Then, the DOTS server activates the "my-ratelimit-list" ACL and replies with a 2.04 (Changed) response to the DOTS client to confirm the successful operation.

As the attack mitigation evolves, the DOTS client may decide to deactivate the rate-limit policy (e.g., upon receipt of a notification status change from 'attack-exceeded-capability' to 'attack-mitigation-in-progress'). Based on the mitigation status conveyed by the DOTS server, the DOTS client can deactivate the rate-limit action. It does so by sending the request shown in Figure 12.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
```

```
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=OopPisZqo4SLv64TLPXrxA"
Uri-Path: "mid=87"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:123::/48"
        ],
        "ietf-dots-signal-control:acl-list": [
          {
            "acl-name": "my-ratelimit-list",
            "activation-type": "deactivate"
          }
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 12: DOTS Signal Channel Mitigation Request to Deactivate a Rate-Limit Filter

5. IANA Considerations

5.1. DOTS Signal Channel CBOR Key Values Subregistry

Per this specification, IANA has registered the following parameters in the "DOTS Signal Channel CBOR Key Values" subregistry within the "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel" registry [Key-Map].

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document(s)
activation-type	52	0	IESG	RFC 9133
ietf-dots-signal-control:acl-list	53	4	IESG	RFC 9133

Table 2

5.2. A New YANG Module

IANA has registered the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

IANA has registered the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

Name: ietf-dots-signal-control
Namespace: urn:ietf:params:xml:ns:yang:ietf-dots-signal-control
Maintained by IANA: N
Prefix: dots-control
Reference: RFC 9133

6. Security Considerations

The security considerations for the DOTS signal channel protocol are discussed in Section 11 of [RFC9132], while those for the DOTS data channel protocol are discussed in Section 10 of [RFC8783]. The following discusses the security considerations that are specific to the DOTS signal channel extension defined in this document.

This specification does not allow the creation of new filtering rules, which is the responsibility of the DOTS data channel. DOTS client domains should be adequately prepared prior to an attack, e.g., by creating filters that will be activated on demand when an attack is detected.

A DOTS client is entitled to access only the resources it creates. In particular, a DOTS client can not tweak filtering rules created by other DOTS clients of the same DOTS client domain. As a reminder, DOTS servers must associate filtering rules with the DOTS client that created these resources. Failure to ensure such association by a DOTS server will have severe impact on DOTS client domains.

A compromised DOTS client can use the filtering control capability to exacerbate an ongoing attack. Likewise, such a compromised DOTS client may abstain from reacting to an ACL conflict notification received from the DOTS server during attacks. These are not new attack vectors, but variations of threats discussed in [RFC9132] and [RFC8783]. DOTS operators should carefully monitor and audit DOTS agents to detect misbehaviors and deter misuses.

7. References

7.1. Normative References

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

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for

the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

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

[RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <https://www.rfc-editor.org/info/rfc8783>.

[RFC8791] Bierman, A., Björklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/info/rfc8791>.

[RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <https://www.rfc-editor.org/info/rfc9132>.

7.2. Informative References

[INTEROP] Nishizuka, K., Shallow, J., and L. Xia, "DOTS Interop test report, IETF 103 Hackathon", November 2018, <https://datatracker.ietf.org/meeting/103/materials/slides-103-dots-interop-report-from-ietf-103-hackathon-00>.

[Key-Map] IANA, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", <https://www.iana.org/assignments/dots>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, DOI 10.17487/RFC8612, May 2019, <https://www.rfc-editor.org/info/rfc8612>.

[RFC8811] Mortensen, A., Ed., Reddy.K, T., Ed., Andreasen, F., Teague, N., and R. Compton, "DDoS Open Threat Signaling (DOTS) Architecture", RFC 8811, DOI 10.17487/RFC8811, August 2020, <https://www.rfc-editor.org/info/rfc8811>.

Acknowledgements

Many thanks to Wei Pan, Xia Liang, Jon Shallow, Dan Wing, Christer Holmberg, Shawn Emery, Tim Chown, Murray Kucherawy, Roman Danyliw, Erik Kline, and Éric Vyncke for the comments.

Thanks to Benjamin Kaduk for the AD review.

Authors' Addresses

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

Email: kaname@nttv6.jp

Mohamed Boucadair
Orange
35000 Rennes
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy.K
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India

Email: kondtir@gmail.com

Takahiko Nagata
Lepidum
Japan

Email: nagata@lepidum.co.jp