DOTS M. Boucadair, Ed.

Internet-Draft Orange

Intended status: Standards Track T. Reddy, Ed.

Expires: August 2, 2020 McAfee

E. Doron

Radware Ltd.

M. Chen

CMCC

January 30, 2020

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry

draft-ietf-dots-telemetry-01

Abstract

This document aims to enrich DOTS signal channel protocol with

various telemetry attributes allowing optimal DDoS attack mitigation.

This document specifies the normal traffic baseline and attack

traffic telemetry attributes a DOTS client can convey to its DOTS

server in the mitigation request, the mitigation status telemetry

attributes a DOTS server can communicate to a DOTS client, and the

mitigation efficacy telemetry attributes a DOTS client can

communicate to a DOTS server. The telemetry attributes can assist

the mitigator to choose the DDoS mitigation techniques and perform

optimal DDoS attack mitigation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the

provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering

Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-

Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months

and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 2, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . 3

2. Terminology . . . . . . . . . . . . . . . . . . . . . . . . . 5

3. DOTS Telemetry: Overview and Purpose . . . . . . . . . . . . 6

4. Generic Considerations . . . . . . . . . . . . . . . . . . . 9

4.1. DOTS Client Identification . . . . . . . . . . . . . . . 9

4.2. DOTS Gateways . . . . . . . . . . . . . . . . . . . . . . 9

4.3. Empty URI Paths . . . . . . . . . . . . . . . . . . . . . 9

4.4. Controlling Configuration Data . . . . . . . . . . . . . 9

4.5. Block-wise Transfer . . . . . . . . . . . . . . . . . . . 10

4.6. YANG Considerations . . . . . . . . . . . . . . . . . . . 10

4.7. A Note About Examples . . . . . . . . . . . . . . . . . . 10

5. Telemetry Operation Paths . . . . . . . . . . . . . . . . . . 10

6. DOTS Telemetry Setup and Configuration . . . . . . . . . . . 11

6.1. Telemetry Configuration . . . . . . . . . . . . . . . . . 12

6.1.1. Retrieve Current DOTS Telemetry Configuration . . . . 12

6.1.2. Convey DOTS Telemetry Configuration . . . . . . . . . 13

6.1.3. Retrieve Installed DOTS Telemetry Configuration . . . 17

6.1.4. Delete DOTS Telemetry Configuration . . . . . . . . . 17

6.2. Total Pipe Capacity . . . . . . . . . . . . . . . . . . . 18

6.2.1. Convey DOTS Client Domain Pipe Capacity . . . . . . . 19

6.2.2. Retrieve DOTS Client Domain Pipe Capacity . . . . . . 23

6.2.3. Delete DOTS Client Domain Pipe Capacity . . . . . . . 23

6.3. Telemetry Baseline . . . . . . . . . . . . . . . . . . . 24

6.3.1. Convey DOTS Client Domain Baseline Information . . . 27

6.3.2. Retrieve Normal Traffic Baseline . . . . . . . . . . 28

6.3.3. Retrieve Normal Traffic Baseline . . . . . . . . . . 28

6.4. Reset Installed Telemetry Setup and Configuration . . . . 28

6.5. Conflict with Other DOTS Clients of the Same Domain . . . 29

7. DOTS Telemetry from Clients to Servers . . . . . . . . . . . 29

7.1. Pre-mitigation DOTS Telemetry Attributes . . . . . . . . 32

7.1.1. Total Attack Traffic . . . . . . . . . . . . . . . . 32

7.1.2. Total Traffic . . . . . . . . . . . . . . . . . . . . 32

7.1.3. Total Attack Connections . . . . . . . . . . . . . . 33

7.1.4. Attack Details . . . . . . . . . . . . . . . . . . . 33

7.2. DOTS Client to Server Mitigation Efficacy DOTS Telemetry

Attributes . . . . . . . . . . . . . . . . . . . . . . . 35

7.2.1. Total Attack Traffic . . . . . . . . . . . . . . . . 35

7.2.2. Attack Details . . . . . . . . . . . . . . . . . . . 35

7.3. Sample Examples . . . . . . . . . . . . . . . . . . . . . 35

7.3.1. Single Pre-Mitigation . . . . . . . . . . . . . . . . 35

7.3.2. Multiple Pre-Mitigations . . . . . . . . . . . . . . 35

7.3.3. Top-Talker of Targets . . . . . . . . . . . . . . . . 35

7.3.4. Top-Talker of Each Target . . . . . . . . . . . . . . 36

8. DOTS Telemetry from Servers to Clients . . . . . . . . . . . 36

8.1. DOTS Server to Client Mitigation Status DOTS Telemetry

Attributes . . . . . . . . . . . . . . . . . . . . . . . 36

8.1.1. Mitigation Status . . . . . . . . . . . . . . . . . . 38

8.2. DOTS Detector to Clients Detection Telemetry . . . . . . 38

9. YANG Module . . . . . . . . . . . . . . . . . . . . . . . . . 38

10. YANG/JSON Mapping Parameters to CBOR . . . . . . . . . . . . 58

11. IANA Considerations . . . . . . . . . . . . . . . . . . . . . 60

11.1. DOTS Signal Channel CBOR Key Values . . . . . . . . . . 60

11.2. DOTS Signal Channel Conflict Cause Codes . . . . . . . . 61

11.3. DOTS Signal Telemetry YANG Module . . . . . . . . . . . 62

12. Security Considerations . . . . . . . . . . . . . . . . . . . 62

13. Contributors . . . . . . . . . . . . . . . . . . . . . . . . 62

14. Acknowledgements . . . . . . . . . . . . . . . . . . . . . . 62

15. References . . . . . . . . . . . . . . . . . . . . . . . . . 63

15.1. Normative References . . . . . . . . . . . . . . . . . . 63

15.2. Informative References . . . . . . . . . . . . . . . . . 64

Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . 65

1. Introduction

Distributed Denial of Service (DDoS) attacks have become more vicious

and sophisticated in almost all aspects of their maneuvers and

malevolent intentions. IT organizations and service providers are

facing DDoS attacks that fall into two broad categories: Network/

Transport layer attacks and Application layer attacks:

o Network/Transport layer attacks target the victim's

infrastructure. These attacks are not necessarily aimed at taking

down the actual delivered services, but rather to eliminate

various network elements (routers, switches, firewalls, transit

links, and so on) from serving legitimate user traffic.

The main method of such attacks is to send a large volume or high

PPS of traffic toward the victim's infrastructure. Typically,

attack volumes may vary from a few 100 Mbps/PPS to 100s of Gbps or

even Tbps. Attacks are commonly carried out leveraging botnets

and attack reflectors for amplification attacks such as NTP

(Network Time Protocol), DNS (Domain Name System), SNMP (Simple

Network Management Protocol), or SSDP (Simple Service Discovery

Protoco).

o Application layer attacks target various applications. Typical

examples include attacks against HTTP/HTTPS, DNS, SIP (Session

Initiation Protocol), or SMTP (Simple Mail Transfer Protocol).

However, all valid applications with their port numbers open at

network edges can be attractive attack targets.

Application layer attacks are considered more complex and hard to

categorize, therefore harder to detect and mitigate efficiently.

To compound the problem, attackers also leverage multi-vectored

attacks. These attacks are assembled from dynamic attack vectors

(Network/Application) and tactics. As such, multiple attack vectors

formed by multiple attack types and volumes are launched

simultaneously towards a victim. Multi-vector attacks are harder to

detect and defend. Multiple and simultaneous mitigation techniques

are needed to defeat such attack campaigns. It is also common for

attackers to change attack vectors right after a successful

mitigation, burdening their opponents with changing their defense

methods.

The ultimate conclusion derived from these real scenarios is that

modern attacks detection and mitigation are most certainly

complicated and highly convoluted tasks. They demand a comprehensive

knowledge of the attack attributes, the targeted normal behavior/

traffic patterns, as well as the attacker's on-going and past

actions. Even more challenging, retrieving all the analytics needed

for detecting these attacks is not simple to obtain with the

industry's current capabilities.

The DOTS signal channel protocol [I-D.ietf-dots-signal-channel] is

used to carry information about a network resource or a network (or a

part thereof) that is under a DDoS attack. Such information is sent

by a DOTS client to one or multiple DOTS servers so that appropriate

mitigation actions are undertaken on traffic deemed suspicious.

Various use cases are discussed in [I-D.ietf-dots-use-cases].

Typically, DOTS clients can be integrated within a DDoS attack

detector, or network and security elements that have been actively

engaged with ongoing attacks. The DOTS client mitigation environment

determines that it is no longer possible or practical for it to

handle these attacks. This can be due to lack of resources or

security capabilities, as derived from the complexities and the

intensity of these attacks. In this circumstance, the DOTS client

has invaluable knowledge about the actual attacks that need to be

handled by the DOTS server. By enabling the DOTS client to share

this comprehensive knowledge of an ongoing attack under specific

circumstances, the DOTS server can drastically increase its abilities

to accomplish successful mitigation. While the attack is being

handled by the DOTS server associated mitigation resources, the DOTS

server has the knowledge about the ongoing attack mitigation. The

DOTS server can share this information with the DOTS client so that

the client can better assess and evaluate the actual mitigation

realized.

In some deployments, DOTS clients can send mitigation hints derived

from attack details to DOTS servers, with the full understanding that

the DOTS server may ignore mitigation hints, as described in

[RFC8612] (Gen-004). Mitigation hints will be transmitted across the

DOTS signal channel, as the data channel may not be functional during

an attack. How a DOTS server is handling normal and attack traffic

attributes, and mitigation hints is implementation-specific.

Both DOTS client and server can benefit this information by

presenting various information in relevant management, reporting, and

portal systems.

This document defines DOTS telemetry attributes the DOTS client can

convey to the DOTS server, and vice versa. The DOTS telemetry

attributes are not mandatory fields. Nevertheless, when DOTS

telemetry attributes are available to a DOTS agent, and absent any

policy, it can signal the attributes in order to optimize the overall

mitigation service provisioned using DOTS. Some of the DOTS

telemetry data is not shared during an attack time.

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].

"DOTS Telemetry" is defined as the collection of attributes that are

used to characterize normal traffic baseline, attacks and their

mitigation measures, and any related information that may help in

enforcing countermeasures. The DOTS Telemetry is an optional set of

attributes that can be signaled in the DOTS signal channel protocol.

The meaning of the symbols in YANG tree diagrams is defined in

[RFC8340].

3. DOTS Telemetry: Overview and Purpose

When signaling a mitigation request, it is most certainly beneficial

for the DOTS client to signal to the DOTS server any knowledge

regarding ongoing attacks. This can happen in cases where DOTS

clients are asking the DOTS server for support in defending against

attacks that they have already detected and/or mitigated. These

actions taken by DOTS clients are referred to as "signaling the DOTS

Telemetry".

If attacks are already detected and categorized by the DOTS client

domain, the DOTS server, and its associated mitigation services, can

proactively benefit this information and optimize the overall service

delivered. It is important to note that DOTS client and server

detection and mitigation approaches can be different, and can

potentially outcome different results and attack classifications.

The DDoS mitigation service treats the ongoing attack details from

the client as hints and cannot completely rely or trust the attack

details conveyed by the DOTS client.

A basic requirement of security operation teams is to be aware and

get visibility into the attacks they need to handle. The DOTS server

security operation teams benefit from the DOTS telemetry, especially

from the reports of ongoing attacks. Even if some mitigation can be

automated, operational teams can use the DOTS telemetry to be

prepared for attack mitigation and to assign the correct resources

(operation staff, networking and mitigation) for the specific

service. Similarly, security operation personnel at the DOTS client

side ask for feedback about their requests for protection.

Therefore, it is valuable for the DOTS server to share DOTS telemetry

with the DOTS client.

Thus mutual sharing of information is crucial for "closing the

mitigation loop" between the DOTS client and server. For the server

side team, it is important to realize that the same attacks that the

DOTS server's mitigation resources are seeing are those that the DOTS

client is asking to mitigate. For the DOTS client side team, it is

important to realize that the DOTS clients receive the required

service. For example: understanding that "I asked for mitigation of

two attacks and my DOTS server detects and mitigates only one...".

Cases of inconsistency in attack classification between DOTS client

and server can be high-lighted, and maybe handled, using the DOTS

telemetry attributes.

In addition, management and orchestration systems, at both DOTS

client and server sides, can potentially use DOTS telemetry as a

feedback to automate various control and management activities

derived from ongoing information signaled.

If the DOTS server's mitigation resources have the capabilities to

facilitate the DOTS telemetry, the DOTS server adopts its protection

strategy and activates the required countermeasures immediately

(automation enabled). The overall results of this adoption are

optimized attack mitigation decisions and actions.

The DOTS telemetry can also be used to tune the DDoS mitigators with

the correct state of the attack. During the last few years, DDoS

attack detection technologies have evolved from threshold-based

detection (that is, cases when all or specific parts of traffic cross

a pre-defined threshold for a certain period of time is considered as

an attack) to an "anomaly detection" approach. In anomaly detection,

the main idea is to maintain rigorous learning of "normal" behavior

and where an "anomaly" (or an attack) is identified and categorized

based on the knowledge about the normal behavior and a deviation from

this normal behavior. Machine learning approaches are used such that

the actual "traffic thresholds" are "automatically calculated" by

learning the protected entity normal traffic behavior during peace

time. The normal traffic characterization learned is referred to as

the "normal traffic baseline". An attack is detected when the

victim's actual traffic is deviating from this normal baseline.

In addition, subsequent activities toward mitigating an attack are

much more challenging. The ability to distinguish legitimate traffic

from attacker traffic on a per packet basis is complex. This

complexity originates from the fact that the packet itself may look

"legitimate" and no attack signature can be identified. The anomaly

can be identified only after detailed statistical analysis. DDoS

attack mitigators use the normal baseline during the mitigation of an

attack to identify and categorize the expected appearance of a

specific traffic pattern. Particularly the mitigators use the normal

baseline to recognize the "level of normality" needs to be achieved

during the various mitigation process.

Normal baseline calculation is performed based on continuous learning

of the normal behavior of the protected entities. The minimum

learning period varies from hours to days and even weeks, depending

on the protected application behavior. The baseline cannot be

learned during active attacks because attack conditions do not

characterize the protected entities' normal behavior.

If the DOTS client has calculated the normal baseline of its

protected entities, signaling this attribute to the DOTS server along

with the attack traffic levels is significantly valuable. The DOTS

server benefits from this telemetry by tuning its mitigation

resources with the DOTS client's normal baseline. The DOTS server

mitigators use the baseline to familiarize themselves with the attack

victim's normal behavior and target the baseline as the level of

normality they need to achieve. Consequently, the overall mitigation

performances obtained are dramatically improved in terms of time to

mitigate, accuracy, false-negative, false-positive, and other

measures.

Mitigation of attacks without having certain knowledge of normal

traffic can be inaccurate at best. This is especially true for

recursive signaling (see Section 3.2.3 in [I-D.ietf-dots-use-cases]).

In addition, the highly diverse types of use-cases where DOTS clients

are integrated also emphasize the need for knowledge of client

behavior. Consequently, common global thresholds for attack

detection practically cannot be realized. Each DOTS client can have

its own levels of traffic and normal behavior. Without facilitating

normal baseline signaling, it may be very difficult for DOTS servers

in some cases to detect and mitigate the attacks accurately:

It is important to emphasize that it is practically impossible for

the server's mitigators to calculate the normal baseline in cases

where they do not have any knowledge of the traffic beforehand.

In addition, baseline learning requires a period of time that

cannot be afforded during active attack.

Of course, this information can provided using out-of-band

mechanisms or manual configuration at the risk to maintain

inaccurate information as the network evolves and "normal"

patterns change. The use of a dynamic and collaborative means

between the DOTS client and server to identify and share key

parameters for the sake of efficient DDoS protection is valuable.

During a high volume attack, DOTS client pipes can be totally

saturated. The DOTS client asks the DOTS server to handle the attack

upstream so that DOTS client pipes return to a reasonable load level

(normal pattern, ideally). At this point, it is essential to ensure

that the mitigator does not overwhelm the DOTS client pipes by

sending back "clean traffic", or what it believes is "clean". This

can happen when the mitigator has not managed to detect and mitigate

all the attacks launched towards the client. In this case, it can be

valuable to clients to signal to server the "Total pipe capacity",

which is the level of traffic the DOTS client domain can absorb from

the upstream network. Dynamic updating of the condition of pipes

between DOTS agents while they are under a DDoS attack is essential.

For example, where multiple DOTS clients share the same

physical connectivity pipes. It is important to note, that the term

"pipe" noted here does not necessary represent physical pipe, but

rather represents the maximum level of traffic that the client can see.

The DOTS server should activate other mechanisms to ensure

it does not allow the client's pipes to be saturated unintentionally. The rate-

limit action defined in [I-D.ietf-dots-data-channel] is a reasonable

candidate to achieve this objective; the client can ask for the type

of traffic (such as ICMP, UDP, TCP port number 80) it prefers to

limit. The rate-limit action can be controlled via the signal-

channel [I-D.ietf-dots-signal-filter-control] even when the pipe is

overwhelmed.

To summarize: timely and effective signaling of up-to-date DOTS

telemetry to all elements involved in the mitigation process is

essential and absolutely improves the overall service effectiveness.

Bi-directional feedback between DOTS agents is required for the

increased awareness of each party, supporting superior and highly

efficient attack mitigation service.

4. Generic Considerations

4.1. DOTS Client Identification

Following the rules in [I-D.ietf-dots-signal-channel], a unique

identifier is generated by a DOTS client to prevent request

collisions.

4.2. DOTS Gateways

DOTS gateways may be located between DOTS clients and servers. The

considerations elaborated in [I-D.ietf-dots-signal-channel] must be

followed. In particular, 'cdid' attribute is used to unambiguously

identify a DOTS client domain.

4.3. Empty URI Paths

Uri-Path parameters with empty values MUST NOT be present in DOTS

telemetry requests.

4.4. Controlling Configuration Data

The DOTS server follows the same considerations discussed in

Section of 4.5.3 of [I-D.ietf-dots-signal-channel] for managing DOTS

telemetry configuration freshness and notification. Likewise, a DOTS

client may control the selection of configuration and non-

configuration data nodes when sending a GET request by means of the

'c' Uri-Query option and following the procedure specified in

Section of 4.4.2 of [I-D.ietf-dots-signal-channel]. These

considerations are not re-iterated in the following sections.

4.5. Block-wise Transfer

DOTS clients can use Block-wise transfer [RFC7959] with the

recommendation detailed in Section 4.4.2 of

[I-D.ietf-dots-signal-channel] to control the size of a response when

the data to be returned does not fit within a single datagram.

DOTS clients can also use Block1 Option in a PUT request (see

Section 2.5 of [RFC7959]) to initiate large transfers, but these Block1 transfers will fail if the inbound “pipe” is running full, so consideration needs to be made to try to fit this PUT into a single transfer, or to separate out the PUT into several discrete PUTs where each of them fits into a single packet.

o NOTE: Add more details.

4.6. YANG Considerations

Messages exchanged between DOTS agents are serialized using Concise

Binary Object Representation (CBOR). CBOR-encoded payloads are used

to carry signal channel-specific payload messages which convey

request parameters and response information such as errors

[I-D.ietf-dots-signal-channel].

This document specifies a YANG module for representing DOTS telemetry

message types (Section 9). All parameters in the payload of the DOTS

signal channel are mapped to CBOR types as specified in Section 10.

4.7. A Note About Examples

Examples are provided for illustration purposes. The document does

not aim to provide a comprehensive list of message examples.

The authoritative reference for validating telemetry messages is the

YANG module (Section 9) and the mapping table established in

Section 10.

5. Telemetry Operation Paths

As discussed in [I-D.ietf-dots-signal-channel], each DOTS operation

is indicated by a path-suffix that indicates the intended operation.

The operation path is appended to the path-prefix to form the URI

used with a CoAP request to perform the desired DOTS operation. The

following telemetry path-suffixes are defined (Table 1):

+-----------------+----------------+-------------+

| Operation | Operation Path | Details |

+-----------------+----------------+-------------+

| Telemetry Setup | /tm-setup | Section 6 |

| Telemetry | /tm | Section 7.1 |

+-----------------+----------------+-------------+

Table 1: DOTS Telemetry Operations

Consequently, the "ietf-dots-telemetry" YANG module defined in this

document augments the "ietf-dots-signal" with two new message types

called "telemetry-setup" and "telemetry". The tree structure of the

"telemetry-setup" message type is shown below (more details are

provided in the following sections about the exact structure of

"telemetry-setup" and "telemetry" message types).

augment /ietf-signal:dots-signal/ietf-signal:message-type:

+--:(telemetry-setup) {dots-telemetry}?

| ...

| +--rw (setup-type)?

| +--:(telemetry-config)

| | ...

| +--:(pipe)

| | ...

| +--:(baseline)

| ...

+--:(telemetry) {dots-telemetry}?

...

Figure 1: New DOTS Message Types (YANG Tree Structure)

6. DOTS Telemetry Setup Configuration

In reference to Figure 1, a DOTS telemetry setup message MUST include

only telemetry-related configuration parameters (Section 6.1) or

information about DOTS client domain pipe capacity (Section 6.2) or

telemetry traffic baseline (Section 6.3). As such, requests that

include a mix of telemetry configuration, pipe capacity, or traffic

baseline MUST be rejected by DOTS servers with a 4.00 (Bad Request).

A DOTS client can reset all installed DOTS telemetry setup

configuration data following the considerations detailed in

Section 6.4.

A DOTS server may detect conflicts when processing requests related

to DOTS client domain pipe capacity or telemetry traffic baseline

with requests from other DOTS clients of the same DOTS client domain.

More details are included in Section 6.5.

DOTS telemetry setup configuration request and response messages

are marked as Confirmable messages.

6.1. Telemetry Configuration

A DOTS client can negotiate with its server(s) a set of telemetry

configuration parameters to be used for telemetry. Such parameters

include:

o Percentile-related measurement parameters

o Measurement units

o Acceptable percentile values

o Server-initiated pre-mitigation telemetry

Section 11.3 of [RFC2330] includes more details about computing

percentiles.

6.1.1. Retrieve Current DOTS Telemetry Configuration

A GET request is used to obtain acceptable and current telemetry

configuration parameters on the DOTS server. This request may

include a 'cdid' Path-URI when the request is relayed by a DOTS

gateway. An example of such request is depicted in Figure 2.

Header: GET (Code=0.01)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Figure 2: GET to Retrieve Current and Acceptable DOTS Telemetry

Configuration

Upon receipt of such request, the DOTS server replies with a 2.05

(Content) response that conveys the current and telemetry parameters

acceptable by the DOTS server. The tree structure of the response

message body is provided in Figure 3. Note that the response

also includes any pipe (Section 6.2) and baseline information

(Section 6.3) maintained by the DOTS server for this DOTS client.

augment /ietf-signal:dots-signal/ietf-signal:message-type:

+--:(telemetry-setup) {dots-telemetry}?

| +--rw telemetry\* [cuid tsid]

| ...

| +--rw (setup-type)?

| +--:(telemetry-config)

| | +--rw current-config

| | | +--rw measurement-interval? interval

| | | +--rw measurement-sample? sample

| | | +--rw low-percentile? percentile

| | | +--rw mid-percentile? percentile

| | | +--rw high-percentile? percentile

| | | +--rw unit-config\* [unit]

| | | | +--rw unit unit

| | | | +--rw unit-status? boolean

| | | +--rw server-initiated-telemetry? boolean

| | +--ro max-config-values

| | | +--ro measurement-interval? interval

| | | +--ro measurement-sample? sample

| | | +--ro low-percentile? percentile

| | | +--ro mid-percentile? percentile

| | | +--ro high-percentile? percentile

| | | +--ro server-initiated-telemetry? boolean

| | +--ro min-config-values

| | | +--ro measurement-interval? interval

| | | +--ro measurement-sample? sample

| | | +--ro low-percentile? percentile

| | | +--ro mid-percentile? percentile

| | | +--ro high-percentile? percentile

| | +--ro supported-units

| | +--ro unit-config\* [unit]

| | +--ro unit unit

| | +--ro unit-status? boolean

| +--:(pipe)

| ...

| +--:(baseline)

| ...

+--:(telemetry) {dots-telemetry}?

+--rw pre-mitigation\* [cuid id]

...

Figure 3: Telemetry Configuration Tree Structure

6.1.2. Convey DOTS Telemetry Configuration

PUT request is used to convey the configuration parameters for the

telemetry data (e.g., low, mid, or high percentile values). For

example, a DOTS client may contact its DOTS server to change the

default percentile values used as baseline for telemetry data.

Figure 3 lists the attributes that can be set by a DOTS client in

such PUT request. An example of a DOTS client that modifies all

percentile reference values is shown in Figure 4.

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=123"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"current-config": {

"low-percentile": 5.00,

"mid-percentile": 65.00,

"high-percentile": 95.00

}

}

]

}

}

Figure 4: PUT to Convey the DOTS Telemetry Configuration

'cuid' is a mandatory Uri-Path parameter for PUT requests.

The following additional Uri-Path parameter is defined:

tsid: Telemetry Setup Identifier is an identifier for the DOTS

telemetry setup and configuration data represented as an

integer. This identifier MUST be generated by DOTS clients.

'tsid' values MUST increase monotonically when a new PUT is

generated by a DOTS client to convey new configuration

parameters for the telemetry.

This is a mandatory attribute.

At least one configurable attribute MUST be present in the PUT

request.

Attributes and Uri-Path parameters with empty values MUST NOT be

present in a request and render the entire request invalid.

The PUT request with a higher numeric 'tsid' value overrides the DOTS

telemetry configuration data installed by a PUT request with a lower

numeric 'tsid' value. To avoid maintaining a long list of 'tsid'

requests for requests carrying telemetry configuration data from a

DOTS client, the lower numeric 'tsid' MUST be automatically deleted

and no longer be available at the DOTS server.

The DOTS server indicates the result of processing the PUT request

using the following response codes:

o If the request is missing a mandatory attribute, does not include

'cuid' or 'tsid' Uri-Path parameters, or contains one or more

invalid or unknown parameters, 4.00 (Bad Request) MUST be returned

in the response.

o If the DOTS server does not find the 'tsid' parameter value

conveyed in the PUT request in its configuration data and if the

DOTS server has accepted the configuration parameters, then a

response code 2.01 (Created) MUST be returned in the response.

o If the DOTS server finds the 'tsid' parameter value conveyed in

the PUT request in its configuration data and if the DOTS server

has accepted the updated configuration parameters, 2.04 (Changed)

MUST be returned in the response.

o If any of the enclosed configurable attribute values are not

acceptable to the DOTS server (Section 6.1.1), 4.22 (Unprocessable

Entity) MUST be returned in the response.

The DOTS client may re-try and send the PUT request with updated

attribute values acceptable to the DOTS server.

Setting 'low-percentile' to '0.00' indicates that the DOTS client is

not interested in receiving low-percentiles. Likewise, setting 'mid-

percentile' (or 'high-percentile') to the same value as 'low-

percentile' (or 'mid-percentile') indicates that the DOTS client is

not interested in receiving mid-percentiles (or high-percentiles).

For example, a DOTS client can send the request depicted in Figure 5

to inform the server that it is interested in receiving only high-

percentiles. This assumes that the client will only use that

percentile type when sharing telemetry data with the server.

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=569"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"current-config": {

"low-percentile": 0.00,

"mid-percentile": 0.00,

"high-percentile": 95.00

}

}

]

}

}

Figure 5: PUT to Disable Low- and Mid-Percentiles

DOTS clients that are interested to receive pre-mitigation telemetry

information from a DOTS server (Section 8) MUST set "server-

initiated-telemetry". An example is shown in Figure 6.

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=569"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"current-config": {

"server-initiated-telemetry": true

}

}

]

}

}

Figure 6: PUT to Enable Pre-mitigation Telemetry from the DOTS server

6.1.3. Retrieve Installed DOTS Telemetry Configuration

A DOTS client may issue a GET message with 'tsid' Uri-Path parameter

to retrieve the current DOTS telemetry configuration. An example of

such request is depicted in Figure 7.

Header: GET (Code=0.01)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=123"

Figure 7: GET to Retrieve Current DOTS Telemetry Configuration

If the DOTS server does not find the 'tsid' Uri-Path value conveyed

in the GET request in its configuration data for the requesting DOTS

client, it MUST respond with a 4.04 (Not Found) error response code.

6.1.4. Delete DOTS Telemetry Configuration

A DELETE request is used to delete the installed DOTS telemetry

configuration data (Figure 8). 'cuid' and 'tsid' are mandatory Uri-

Path parameters for such DELETE requests.

Header: DELETE (Code=0.04)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=123"

Figure 8: Delete Telemetry Configuration

If the DELETE request does not include 'cuid' and 'tsid' parameters,

the DOTS server MUST reply with a 4.00 (Bad Request).

The DOTS server resets the DOTS telemetry configuration back to the

default values and acknowledges a DOTS client's request to remove the

DOTS telemetry configuration using 2.02 (Deleted) response code. A

2.02 (Deleted) Response Code is returned even if the 'tsid' parameter

value conveyed in the DELETE request does not exist in its

configuration data before the request.

6.2. Total Pipe Capacity

A DOTS client can communicate to its server(s) its DOTS client domain

pipe information. The tree structure of the pipe information is

shown in Figure 9.

augment /ietf-signal:dots-signal/ietf-signal:message-type:

+--:(telemetry-setup) {dots-telemetry}?

| +--rw telemetry\* [cuid tsid]

| +--rw cuid string

| +--rw cdid? string

| +--rw tsid uint32

| +--rw (setup-type)?

| +--:(telemetry-config)

| | ...

| +--:(pipe)

| | +--rw total-pipe-capacity\* [link-id unit]

| | +--rw link-id nt:link-id

| | +--rw capacity uint64

| | +--rw unit unit

| +--:(baseline)

| ...

+--:(telemetry) {dots-telemetry}?

+--rw pre-mitigation\* [cuid id]

...

Figure 9: Pipe Tree Structure

A DOTS client domain pipe is defined as a list of limits of

(incoming) traffic volume (total-pipe-capacity") that can be

forwarded over ingress interconnection links for a DOTS client domain.

Each of these links is identified with a "link-id" [RFC8345].

This limit can be expressed in packets per second (PPS) or kilo

packets per second (Kpps) and Bits per Second (BPS), and in kilobytes

per second or megabytes per second or gigabytes per second. The unit

used by a DOTS client when conveying pipe information is captured in

"unit" attribute.

6.2.1. Convey DOTS Client Domain Pipe Capacity

Similar considerations to those specified in Section 6.1.2 are

followed with one exception:

The relative order of two PUT requests carrying DOTS client domain

pipe attributes from a DOTS client is determined by comparing

their respective 'tsid' values. If such two requests have

overlapping "link-id" and "unit", the PUT request with higher

numeric 'tsid' value will override the request with a lower

numeric 'tsid' value. The overlapped lower numeric 'tsid' MUST be

automatically deleted and no longer be available.

DOTS clients SHOULD minimize the number of active "tsids" used for

pipe information. Typically, in order to avoid maintaining a long

list of "tsids" for pipe information, it is RECOMMENDED that DOTS

clients include in any request to update information related to a given

link the information of other links (already communicated using a

lower 'tsid' value). Doing so, this update request will override

these existing requests and hence optimize the number of 'tsid"

request per DOTS client.

o Note: This assumes that all link information can fit in one single

message.

For example, a DOTS client managing a single homed domain (Figure 10)

can send a PUT request (shown in Figure 11) to communicate the

capacity of "link1" used to connected its ISP.

,--,--,--. ,--,--,--.

,-' `-. ,-' `-.

( DOTS Client )=====( ISP#A )

`-. Domain ,-' link1 `-. ,-'

`--'--'--' `--'--'--'

Figure 10: Single Homed DOTS Client Domain

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=457"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"total-pipe-capacity": [

{

"link-id": "link1",

"capacity": 500,

"unit": "megabytes-ps"

}

]

}

]

}

}

Figure 11: Example of a PUT Request to Convey Pipe Information

(Single Homed)

Now consider that the DOTS client domain was upgraded to connect to

an additional ISP (ISP#B of Figure 12), the DOTS client can inform

the DOTS server about this update by sending the PUT request depicted

in Figure 13. This request includes also information related to

"link1" even if that link is not upgraded. Upon receipt of this

request, the DOTS server removes the request with "tsid=457" and

updates its configuration base to maintain two links (link#1 and

link#2).

,--,--,--.

,-' `-.

( ISP#B )

`-. ,-'

`--'--'--'

||

|| link2

,--,--,--. ,--,--,--.

,-' `-. ,-' `-.

( DOTS Client )=====( ISP#A )

`-. Domain ,-' link1 `-. ,-'

`--'--'--' `--'--'--'

Figure 12: Multi-Homed DOTS Client Domain

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=458"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"total-pipe-capacity": [

{

"link-id": "link1",

"capacity": 500,

"unit": "megabytes-ps"

},

{

"link-id": "link2",

"capacity": 500,

"unit": "megabytes-ps"

}

]

}

]

}

}

Figure 13: Example of a PUT Request to Convey Pipe Information

(Multi-Homed)

A DOTS client can delete a link by sending a PUT request with the

capacity" attribute set to "0" if other links are still active for

the same DOTS client domain (see Section 6.2.3 for other delete

cases). For example, if a DOTS client domain re-homes (that is, it

changes its ISP), the DOTS client can inform the DOTS server about

this update (e.g., from the network configuration in Figure 10 to the

one shown in Figure 14) by sending the PUT request depicted in

Figure 15. Upon receipt of this request, the DOTS server removes

"link1" from its configuration bases for this DOTS client domain.

,--,--,--.

,-' `-.

( ISP#B )

`-. ,-'

`--'--'--'

||

|| link2

,--,--,--.

,-' `-.

( DOTS Client )

`-. Domain ,-'

`--'--'--'

Figure 14: Multi-Homed DOTS Client Domain

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=459"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry-setup": {

"telemetry": [

{

"total-pipe-capacity": [

{

"link-id": "link1",

"capacity": 0,

"unit": "megabytes-ps"

},

{

"link-id": "link2",

"capacity": 500,

"unit": "megabytes-ps"

}

]

}

]

}

}

Figure 15: Example of a PUT Request to Convey Pipe Information

(Multi-Homed)

6.2.2. Retrieve DOTS Client Domain Pipe Capacity

A GET request with 'tsid' Uri-Path parameter is used to retrieve a

specific installed DOTS client domain pipe related information. The

same procedure as defined in (Section 6.1.3) is followed.

To retrieve all pipe information bound to a DOTS client, the DOTS

client proceeds as specified in Section 6.1.1.

6.2.3. Delete DOTS Client Domain Pipe Capacity

A DELETE request is used to delete the installed DOTS client domain

pipe related information. The same procedure as defined

in (Section 6.1.4) is followed.

6.3. Telemetry Baseline

A DOTS client can communicate to its server(s) its normal traffic

baseline and total connections capacity:

Total Traffic Normal Baseline: By default, the low percentile (10th

percentile), mid percentile (50th percentile), high percentile

(90th percentile), and peak values (100th percentile) of "Total

traffic normal baselines" measured in packets per second (PPS) or

kilo packets per second (Kpps) and Bits per Second (BPS), and

kilobytes per second or megabytes per second or gigabytes per

second. For example, 90th percentile says that 90% of the time,

the total normal traffic is below the limit specified.

The traffic normal baseline is represented for a target and is

transport-protocol specific.

If the DOTS client negotiated percentile values and units

(Section 6.1), these negotiated values will be used instead of the

default ones.

Total Connections Capacity: If the target is subjected to resource

consuming DDoS attacks, the following optional attributes for the

target per transport-protocol are useful to detect resource

consuming DDoS attacks:

\* The maximum number of simultaneous connections that are allowed

to the target. The threshold is transport-protocol specific

because the target could support multiple protocols.

\* The maximum number of simultaneous connections that are allowed

to the target per client.

\* The maximum number of simultaneous embryonic connections that

are allowed to the target. The term "embryonic connection"

refers to a connection whose connection handshake is not

finished and embryonic connection is only possible in

connection-oriented transport protocols like TCP or SCTP.

\* The maximum number of simultaneous embryonic connections that

are allowed to the target per client.

\* The maximum number of connections allowed per second to the

target.

\* The maximum number of connections allowed per second to the

target per client.

\* The maximum number of requests allowed per second to the

target.

\* The maximum number of requests allowed per second to the target

per client.

\* The maximum number of partial requests allowed per second to

the target.

\* The maximum number of partial requests allowed per second to

the target per client.

The tree structure of the baseline is shown in Figure 16.

augment /ietf-signal:dots-signal/ietf-signal:message-type:

+--:(telemetry-setup) {dots-telemetry}?

| +--rw telemetry\* [cuid tsid]

| +--rw cuid string

| +--rw cdid? string

| +--rw tsid uint32

| +--rw (setup-type)?

| +--:(telemetry-config)

| | ...

| +--:(pipe)

| | ...

| +--:(baseline)

| +--rw baseline\* [id]

| +--rw id uint32

| +--rw target-prefix\* inet:ip-prefix

| +--rw target-port-range\* [lower-port]

| | +--rw lower-port inet:port-number

| | +--rw upper-port? inet:port-number

| +--rw target-protocol\* uint8

| +--rw target-fqdn\* inet:domain-name

| +--rw target-uri\* inet:uri

| +--rw total-traffic-normal-baseline\* [unit protocol]

| | +--rw unit unit

| | +--rw protocol uint8

| | +--rw low-percentile-g? yang:gauge64

| | +--rw mid-percentile-g? yang:gauge64

| | +--rw high-percentile-g? yang:gauge64

| | +--rw peak-g? yang:gauge64

| +--rw total-connection-capacity\* [protocol]

| +--rw protocol uint8

| +--rw connection? uint64

| +--rw connection-client? uint64

| +--rw embryonic? uint64

| +--rw embryonic-client? uint64

| +--rw connection-ps? uint64

| +--rw connection-client-ps? uint64

| +--rw request-ps? uint64

| +--rw request-client-ps? uint64

| +--rw partial-request-ps? uint64

| +--rw partial-request-client-ps? uint64

+--:(telemetry) {dots-telemetry}?

+--rw pre-mitigation\* [cuid id]

...

Figure 16: Telemetry Baseline Tree Structure

6.3.1. Convey DOTS Client Domain Baseline Information

Similar considerations to those specified in Section 6.1.2 are

followed with one exception:

The relative order of two PUT requests carrying DOTS client domain

baseline attributes from a DOTS client is determined by comparing

their respective 'tsid' values. If such two requests have

overlapping targets, the PUT request with higher numeric 'tsid'

value will override the request with a lower numeric 'tsid' value.

The overlapped lower numeric 'tsid' MUST be automatically deleted

and no longer be available.

Two PUT requests from a DOTS client have overlapping targets if there

is a common IP address, IP prefix, FQDN, or URI.

DOTS clients SHOULD minimize the number of active "tsids" used for

baseline information. Typically, in order to avoid maintaining a

long list of "tsids" for baseline information, it is RECOMMENDED that

DOTS clients include in a request to update information related to a

given target, the information of other targets (already communicated

using a lower 'tsid' value) (assuming this fits within one single

datagram). This update request will override these existing requests

and hence optimize the number of 'tsid' request per DOTS client.

If no target clause in included in the request, this is an indication

that the baseline information applies for the DOTS client domain as a

whole.

An example of a PUT request to convey the baseline information is

shown in Figure 17.

Header: PUT (Code=0.03)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Uri-Path: "tsid=126"

Content-Format: "application/dots+cbor"

{

"ietf-dots-signal-channel:telemetry": {

"baseline": {

"id": 1,

"target-prefix": [

"2001:db8:6401::1/128",

"2001:db8:6401::2/128"

],

"total-traffic-normal-baseline": {

"unit": "megabytes-ps",

"protocol": 6,

"peak-g": "50"

}

}

}

}

Figure 17: PUT to Convey the DOTS Traffic Baseline

6.3.2. Retrieve Normal Traffic Baseline

A GET request with 'tsid' Uri-Path parameter is used to retrieve a

specific installed DOTS client domain baseline traffic information.

The same procedure as defined in (Section 6.1.3) is

followed.

To retrieve all baseline information bound to a DOTS client, the DOTS

client proceeds as specified in Section 6.1.1.

6.3.3. Retrieve Normal Traffic Baseline

A DELETE request is used to delete the installed DOTS client domain

normal traffic baseline. The same procedure as defined in

(Section 6.1.4) is followed.

6.4. Reset Installed Telemetry Setup

Upon bootstrapping (or reboot or any other event that may alter the environment,

a DOTS client MAY send a DELETE request to set the telemetry

parameters to default values. Such a request does not include any

'tsid'. An example of such request is depicted in Figure 18.

Header: DELETE (Code=0.04)

Uri-Path: ".well-known"

Uri-Path: "dots"

Uri-Path: "tm-setup"

Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"

Figure 18: Delete Telemetry Configuration

6.5. Conflict with Other DOTS Clients of the Same Domain

A DOTS server may detect conflicts between requests to convey pipe

and baseline information received from DOTS clients of the same DOTS

client domain. 'conflict-information' is used to report the conflict

to the DOTS client following similar conflict handling discussed in

Section 4.4.1 of [I-D.ietf-dots-signal-channel]. The conflict cause

can be set to one of these values:

1: Overlapping targets (already defined in

[I-D.ietf-dots-signal-channel]).

TBA: Overlapping pipe scope (see Section 11).

7. DOTS Telemetry from Clients to Servers

There are two broad types of DDoS attacks, one is bandwidth consuming

attack, the other is target resource consuming attack. This section

outlines the set of DOTS telemetry attributes (Section 7.1) that

covers both the types of attacks. The ultimate objective of these

attributes is to allow for the complete knowledge of attacks and the

various particulars that can best characterize attacks.

The description and motivation behind each attribute are presented in

Section 3. DOTS telemetry attributes are optionally signaled and

therefore MUST NOT be treated as mandatory fields in the DOTS signal

channel protocol.

The "ietf-dots-telemetry" YANG module augments the "ietf-dots-signal"

with two new message types called "telemetry-setup" and "telemetry".

The tree structure of the "telemetry" message type is shown below:

augment /ietf-signal:dots-signal/ietf-signal:message-type:

+--:(telemetry-setup) {dots-telemetry}?

| +--rw telemetry\* [cuid tsid]

| ...

+--:(telemetry) {dots-telemetry}?

+--rw pre-mitigation\* [cuid id]

+--rw cuid string

+--rw cdid? string

+--rw id uint32

+--rw target

| +--rw target-prefix\* inet:ip-prefix

| +--rw target-port-range\* [lower-port]

| | +--rw lower-port inet:port-number

| | +--rw upper-port? inet:port-number

| +--rw target-protocol\* uint8

| +--rw target-fqdn\* inet:domain-name

| +--rw target-uri\* inet:uri

+--rw total-traffic\* [unit protocol]

| +--rw unit unit

| +--rw protocol uint8

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-traffic\* [unit protocol]

| +--rw unit unit

| +--rw protocol uint8

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-connection

| +--rw low-percentile-l\* [protocol]

| | +--rw protocol uint8

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw mid-percentile-l\* [protocol]

| | +--rw protocol uint8

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw high-percentile-l\* [protocol]

| | +--rw protocol uint8

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw peak-l\* [protocol]

| +--rw protocol uint8

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw attack-detail

+--rw id? uint32

+--rw attack-id? string

+--rw attack-name? string

+--rw attack-severity? attack-severity

+--rw start-time? uint64

+--rw end-time? uint64

+--rw source-count

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw top-talker

+--rw source-prefix\* [source-prefix]

+--rw spoofed-status? boolean

+--rw source-prefix inet:ip-prefix

+--rw total-attack-traffic\* [unit]

| +--rw unit unit

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-connection

+--rw low-percentile-l\* [protocol]

| +--rw protocol uint8

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw mid-percentile-l\* [protocol]

| +--rw protocol uint8

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw high-percentile-l\* [protocol]

| +--rw protocol uint8

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw peak-l\* [protocol]

+--rw protocol uint8

+--rw connection? yang:gauge64

+--rw embryonic? yang:gauge64

+--rw connection-ps? yang:gauge64

+--rw request-ps? yang:gauge64

+--rw partial-request-ps? yang:gauge64

7.1. Pre-mitigation DOTS Telemetry Attributes

The pre-mitigation telemetry attributes are indicated by the path-

suffix '/tm'. The '/tm' is appended to the path-prefix to form the

URI used with a CoAP request to signal the DOTS telemetry. The

following pre-mitigation telemetry attributes can be signaled from

DOTS clients to DOTS servers.

o DISCUSSION NOTES: (1) Some telemetry can be communicated using

DOTS data channel. (2) Evaluate the risk of fragmentation,. Some

of the information is not specific to each mitigation request. (3)

Should we define other configuration parameters to be controlled

by a DOTS client, e.g., Indicate a favorite measurement unit?

Indicate a minimum notification interval?

7.1.1. Total Attack Traffic

The total attack traffic can be identified by the DOTS client

domain's DDoS Mitigation System (DMS) or DDoS Detector. The low

percentile (10th percentile), mid percentile (50th percentile), high

percentile (90th percentile) and peak values of total attack traffic

measured in packets per second (PPS) or kilo packets per second

(Kpps) and Bits per Second (BPS), and kilobytes per second or

megabytes per second or gigabytes per second. The total attack

traffic is represented for a target and is transport-protocol

specific.

7.1.2. Total Traffic

The low percentile (10th percentile), mid percentile (50th

percentile), high percentile (90th percentile) and peak values of

total traffic during a DDoS attack measured in packets per second

(PPS) or kilo packets per second (Kpps) and Bits per Second (BPS),

and kilobytes per second or megabytes per second gigabytes per

second. The total traffic is represented for a target and is

transport-protocol specific.

7.1.3. Total Attack Connections

If the target is subjected to resource consuming DDoS attack, the low

percentile (10th percentile), mid percentile (50th percentile), high

percentile (90th percentile) and peak values of following optional

attributes for the target per transport-protocol are included to

represent the attack characteristics:

o The number of simultaneous attack connections to the target

server.

o The number of simultaneous embryonic connections to the target

server.

o The number of attack connections per second to the target server.

o The number of attack requests to the target server.

7.1.4. Attack Details

The attack details need to cover well-known and common attacks (such

as a SYN Flood) along with new emerging or vendor-specific attacks.

The following new fields describing the on-going attack are

discussed:

vendor-id: Vendor ID is a security vendor's Enterprise Number as

registered with IANA [Enterprise-Numbers]. It is a four-byte

integer value.

This is a mandatory sub-attribute.

attack-id: Unique identifier assigned by the vendor for the attack.

This is a mandatory sub-attribute.

attack-name: Textual representation of attack description. Natural

Language Processing techniques (e.g., word embedding) can possibly

be used to map the attack description to an attack type. Textual

representation of attack solves two problems (a) avoids the need

to create mapping tables manually between vendors (2) Avoids the

need to standardize attack types which keep evolving.

This is a mandatory sub-attribute

attack-severity: Attack severity. Emergency (1), critical (2) and

alert (3).

This is an optional sub-attribute

start-time: The time the attack started. The attack start time is

expressed in seconds relative to 1970-01-01T00:00Z in UTC time

(Section 2.4.1 of [RFC7049]). The CBOR encoding is modified so

that the leading tag 1 (epoch-based date/time) MUST be omitted.

This is a mandatory sub-attribute

end-time: The time the attack-id attack ended. The attack

end time is expressed in seconds relative to 1970-01-01T00:00Z in

UTC time (Section 2.4.1 of [RFC7049]). The CBOR encoding is

modified so that the leading tag 1 (epoch-based date/time) MUST be

omitted.

This is an optional sub-attribute

The following existing fields are re-defined describing the on-going

attack are discussed:

o The target resource is identified using the attributes 'target-

prefix', 'target-port-range', 'target-protocol', 'target-

fqdn','target-uri', or 'alias-name' defined in the base DOTS

signal channel protocol and at least one of the attributes

'target-prefix', 'target-fqdn','target-uri', or 'alias-name' MUST

be present in the attack details.

A. If the target is subjected to bandwidth consuming attack, the

attributes representing the low percentile (10th percentile),

mid percentile (50th percentile), high percentile (90th

percentile) and peak values of the attack-id attack traffic

measured in packets per second (PPS) or kilo packets per

second (Kpps) and Bits per Second (BPS), and kilobytes per

second or megabytes per second or gigabytes per second are

included.

B. If the target is subjected to resource consuming DDoS attacks,

the same attributes defined for Section 7.1.3 are applicable

for representing the attack.

This is an optional sub-attribute.

o A count of sources involved in the attack targeting the victim and

a list of top talkers among those sources. The top talkers are

represented using the 'source-prefix' defined in

[I-D.ietf-dots-signal-call-home]. If the top talkers are spoofed

IP addresses (e.g., reflection attacks) or not. If the target is

subjected to bandwidth consuming attack, the attack traffic from

each of the top talkers represented in the low percentile (10th

percentile), mid percentile (50th percentile), high percentile

(90th percentile) and peak values of traffic measured in packets

per second (PPS) or kilo packets per second (Kpps) and Bits per

Second (BPS), and kilobytes per second or megabytes per second

gigabytes per second. If the target is subjected to resource

consuming DDoS attacks, the same attributes defined for

Section 7.1.3 are applicable here for representing the attack per

talker. This is an optional sub-attribute.

7.2. DOTS Client to Server Mitigation Efficacy DOTS Telemetry

Attributes

The mitigation efficacy telemetry attributes can be signaled from the

DOTS client to the DOTS server as part of the periodic mitigation

efficacy updates to the server (Section 5.3.4 of

[I-D.ietf-dots-signal-channel]).

7.2.1. Total Attack Traffic

The low percentile (10th percentile), mid percentile (50th

percentile), high percentile (90th percentile), and peak values of

total attack traffic the DOTS client still sees during the active

mitigation service measured in packets per second (PPS) or kilo

packets per second (Kpps) and Bits per Second (BPS), and kilobytes

per second or megabytes per second or gigabytes per second.

7.2.2. Attack Details

The overall attack details as observed from the DOTS client

perspective during the active mitigation service. The same

attributes defined in Section 7.1.4 are applicable here.

7.3. Sample Examples

7.3.1. Single Pre-Mitigation

<<>>

7.3.2. Multiple Pre-Mitigations

<<multiple mitigation-ids are used >>

7.3.3. Top-Talker of Targets

<<A server can aggregate top-talkers for all targets of a domain, or

when justified, send specific information (including top-talkers) per

individual targets. >>

<<several target victim (target) addresses should be included in the

target-prefix\*.>>

7.3.4. Top-Talker of Each Target

<<Each target victim (target) address should be included in the list

of target-prefix\* in each pre-mitigation, and several pre-mitigations

should be included in the pre-mitigation\*.>>

8. DOTS Telemetry from Servers to Clients

8.1. DOTS Server to Client Mitigation Status DOTS Telemetry Attributes

The mitigation status telemetry attributes can be signaled from the

DOTS server to the DOTS client as part of the periodic mitigation

status update (Section 5.3.3 of [I-D.ietf-dots-signal-channel]). In

particular, DOTS clients can receive asynchronous notifications of

the attack details from DOTS servers using the Observe option defined

in [RFC7641].

The "ietf-dots-telemetry" YANG module augments the "mitigation-scope"

type message defined in "ietf-dots-signal" with telemetry data as

depicted in following tree structure:

augment /ietf-signal:dots-signal/ietf-signal:message-type

/ietf-signal:mitigation-scope/ietf-signal:scope:

+--rw total-traffic\* [unit protocol] {dots-telemetry}?

| +--rw unit unit

| +--rw protocol uint8

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-traffic\* [unit] {dots-telemetry}?

| +--rw unit unit

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-connection {dots-telemetry}?

| +--rw low-percentile-c

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw mid-percentile-c

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw high-percentile-c

| | +--rw connection? yang:gauge64

| | +--rw embryonic? yang:gauge64

| | +--rw connection-ps? yang:gauge64

| | +--rw request-ps? yang:gauge64

| | +--rw partial-request-ps? yang:gauge64

| +--rw peak-c

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw attack-detail {dots-telemetry}?

+--rw id? uint32

+--rw attack-id? string

+--rw attack-name? string

+--rw attack-severity? attack-severity

+--rw start-time? uint64

+--rw end-time? uint64

+--rw source-count

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw top-talker

+--rw source-prefix\* [source-prefix]

+--rw spoofed-status? boolean

+--rw source-prefix inet:ip-prefix

+--rw total-attack-traffic\* [unit]

| +--rw unit unit

| +--rw low-percentile-g? yang:gauge64

| +--rw mid-percentile-g? yang:gauge64

| +--rw high-percentile-g? yang:gauge64

| +--rw peak-g? yang:gauge64

+--rw total-attack-connection

+--rw low-percentile-c

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw mid-percentile-c

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw high-percentile-c

| +--rw connection? yang:gauge64

| +--rw embryonic? yang:gauge64

| +--rw connection-ps? yang:gauge64

| +--rw request-ps? yang:gauge64

| +--rw partial-request-ps? yang:gauge64

+--rw peak-c

+--rw connection? yang:gauge64

+--rw embryonic? yang:gauge64

+--rw connection-ps? yang:gauge64

+--rw request-ps? yang:gauge64

+--rw partial-request-ps? yang:gauge64

8.1.1. Mitigation Status

As defined in [RFC8612], the actual mitigation activities can include

several countermeasure mechanisms. The DOTS server SHOULD signal the

current operational status to each relevant countermeasure. A list

of attacks detected by each countermeasure.

The same attributes defined for Section 7.1.4 are applicable for

describing the attacks detected and mitigated.

8.2. DOTS Detector to Clients Detection Telemetry

The attack details can also be signaled from DOTS servers to DOTS

clients. For example, the DOTS server co-located with a DDoS

detector collects monitoring information from the target network,

identifies DDoS attack using statistical analysis or deep learning

techniques, and signals the attack details to the DOTS client.

The DOTS client can use the attack details to decide whether to

trigger a DOTS mitigation request or not. Furthermore, the security

operation personnel at the DOTS client domain can use the attack

details to determine the protection strategy and select the

appropriate DOTS server for mitigating the attack.

<<to be further discussed>>

9. YANG Module

This module uses types defined in [RFC6991].

<CODE BEGINS> file "ietf-dots-telemetry@2020-01-23.yang"

module ietf-dots-telemetry {

yang-version 1.1;

namespace "urn:ietf:params:xml:ns:yang:ietf-dots-telemetry";

prefix dots-telemetry;

import ietf-dots-signal-channel {

prefix ietf-signal;

reference

"RFC SSSS: Distributed Denial-of-Service Open Threat

Signaling (DOTS) Signal Channel Specification";

}

import ietf-dots-data-channel {

prefix ietf-data;

reference

"RFC DDDD: Distributed Denial-of-Service Open Threat

Signaling (DOTS) Data Channel Specification";

}

import ietf-yang-types {

prefix yang;

reference

"Section 3 of RFC 6991";

}

import ietf-inet-types {

prefix inet;

reference

"Section 4 of RFC 6991";

}

import ietf-network-topology {

prefix nt;

reference

"Section 6.2 of RFC 8345: A YANG Data Model for Network

Topologies";

}

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: Mohamed Boucadair

<mailto:mohamed.boucadair@orange.com>

Author: Konda, Tirumaleswar Reddy

<mailto:TirumaleswarReddy\_Konda@McAfee.com>";

description

"This module contains YANG definitions for the signaling

of DOTS telemetry exchanged between a DOTS client and

a DOTS server, by means of the DOTS signal channel.

Copyright (c) 2020 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 2020-01-23 {

description

"Initial revision.";

reference

"RFC XXXX: Distributed Denial-of-Service Open Threat

Signaling (DOTS) Telemetry";

}

feature dots-telemetry {

description

"This feature means that the DOTS signal channel is able

to convey DOTS telemetry data between DOTS clients and

servers.";

}

typedef attack-severity {

type enumeration {

enum emergency {

value 1;

description

"The attack is severe: emergency.";

}

enum critical {

value 2;

description

"The attack is critical.";

}

enum alert {

value 3;

description

"This is an alert.";

}

}

description

"Enumeration for attack severity.";

}

typedef unit {

type enumeration {

enum pps {

value 1;

description

"Packets per second (PPS).";

}

enum kilo-pps {

value 2;

description

"Kilo packets per second (Kpps).";

}

enum bps {

value 3;

description

"Bits per Second (BPS).";

}

enum kilobytes-ps {

value 4;

description

"Kilobytes per second.";

}

enum megabytes-ps {

value 5;

description

"Megabytes per second.";

}

enum gigabytes-ps {

value 6;

description

"Gigabytes per second.";

}

}

description

"Enumeration to indicate which unit is used.";

}

typedef interval {

type enumeration {

enum hour {

value 1;

description

"Hour.";

}

enum day {

value 2;

description

"Day.";

}

enum week {

value 3;

description

"Week.";

}

enum month {

value 4;

description

"Month.";

}

}

description

"Enumeration to indicate the overall measurement period.";

}

typedef sample {

type enumeration {

enum second {

value 1;

description

"Second.";

}

enum 5-seconds {

value 2;

description

"5 seconds.";

}

enum 30-seconds {

value 3;

description

"30 seconds.";

}

enum minute {

value 4;

description

"One minute.";

}

enum 5-minutes {

value 5;

description

"5 minutes.";

}

enum 10-minutes {

value 6;

description

"10 minutes.";

}

enum 30-minutes {

value 7;

description

"30 minutes.";

}

enum hour {

value 8;

description

"One hour.";

}

}

description

"Enumeration to indicate the measurement perdiod.";

}

typedef percentile {

type decimal64 {

fraction-digits 2;

}

description

"The nth percentile of a set of data is the

value at which n percent of the data is below it.";

}

grouping percentile-config {

description

"Configuration of low, mid, and high percentile values.";

leaf measurement-interval {

type interval;

description

"Defines the period on which percentiles are computed.";

}

leaf measurement-sample {

type sample;

description

"Defines the time distribution for measuring

values that are used to compute percentiles..";

}

leaf low-percentile {

type percentile;

default "10.00";

description

"Low percentile. If set to '0', this means low-percentiles

are disabled.";

}

leaf mid-percentile {

type percentile;

must '. >= ../low-percentile' {

error-message

"The mid-percentile must be greater than

or equal to the low-percentile.";

}

default "50.00";

description

"Mid percentile. If set to the same value as low-percentiles,

this means mid-percentiles are disabled.";

}

leaf high-percentile {

type percentile;

must '. >= ../mid-percentile' {

error-message

"The high-percentile must be greater than

or equal to the mid-percentile.";

}

default "90.00";

description

"High percentile. If set to the same value as mid-percentiles,

this means high-percentiles are disabled.";

}

}

grouping percentile {

description

"Generic grouping for percentile.";

leaf low-percentile-g {

type yang:gauge64;

description

"Low traffic.";

}

leaf mid-percentile-g {

type yang:gauge64;

description

"Mid percentile.";

}

leaf high-percentile-g {

type yang:gauge64;

description

"High percentile.";

}

leaf peak-g {

type yang:gauge64;

description

"Peak";

}

}

grouping unit-config {

description

"Generic grouping for unit configuration.";

list unit-config {

key "unit";

description

"Controls which units are allowed when sharing telemetry

data.";

leaf unit {

type unit;

description

"The traffic can be measured in packets per

second (PPS) or kilo packets per second (Kpps) and Bits per

Second (BPS), and kilobytes per second or megabytes per second

or gigabytes per second.";

}

leaf unit-status {

type boolean;

description

"Enable/disable the use of the measurement unit.";

}

}

}

grouping traffic-unit {

description

"Grouping of traffic as a function of measurement unit.";

leaf unit {

type unit;

description

"The traffic can be measured in packets per

second (PPS) or kilo packets per second (Kpps) and Bits per

Second (BPS), and kilobytes per second or megabytes per second

or gigabytes per second.";

}

uses percentile;

}

grouping traffic-unit-protocol {

description

"Grouping of traffic of a given transport protocol as

a function of measurement unit.";

leaf unit {

type unit;

description

"The traffic can be measured in packets per

second (PPS) or kilo packets per second (Kpps) and Bits per

Second (BPS), and kilobytes per second or megabytes per second

or gigabytes per second.";

}

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.

For example, this field contains 6 for TCP,

17 for UDP, 33 for DCCP, or 132 for SCTP.";

}

uses percentile;

}

grouping total-connection-capacity {

description

"Total Connections Capacity. If the target is subjected

to resource consuming DDoS attack, these attributes are

useful to detect resource consuming DDoS attacks";

leaf connection {

type uint64;

description

"The maximum number of simultaneous connections that

are allowed to the target server. The threshold is

transport-protocol specific because the target server

could support multiple protocols.";

}

leaf connection-client {

type uint64;

description

"The maximum number of simultaneous connections that

are allowed to the target server per client.";

}

leaf embryonic {

type uint64;

description

"The maximum number of simultaneous embryonic connections

that are allowed to the target server. The term 'embryonic

connection' refers to a connection whose connection handshake

is not finished and embryonic connection is only possible in

connection-oriented transport protocols like TCP or SCTP.";

}

leaf embryonic-client {

type uint64;

description

"The maximum number of simultaneous embryonic connections

that are allowed to the target server per client.";

}

leaf connection-ps {

type uint64;

description

"The maximum number of connections allowed per second

to the target server.";

}

leaf connection-client-ps {

type uint64;

description

"The maximum number of connections allowed per second

to the target server per client.";

}

leaf request-ps {

type uint64;

description

"The maximum number of requests allowed per second

to the target server.";

}

leaf request-client-ps {

type uint64;

description

"The maximum number of requests allowed per second

to the target server per client.";

}

leaf partial-request-ps {

type uint64;

description

"The maximum number of partial requests allowed per

second to the target server.";

}

leaf partial-request-client-ps {

type uint64;

description

"The maximum number of partial requests allowed per

second to the target server per client.";

}

}

grouping connection {

description

"A set of attributes which represent the attack

characteristics";

leaf connection {

type yang:gauge64;

description

"The number of simultaneous attack connections to

the target server.";

}

leaf embryonic {

type yang:gauge64;

description

"The number of simultaneous embryonic connections to

the target server.";

}

leaf connection-ps {

type yang:gauge64;

description

"The number of attack connections per second to

the target server.";

}

leaf request-ps {

type yang:gauge64;

description

"The number of attack requests per second to

the target server.";

}

leaf partial-request-ps {

type yang:gauge64;

description

"The number of attack partial requests to

the target server.";

}

}

grouping connection-percentile {

description

"Total attack connections.";

container low-percentile-c {

description

"Low percentile of attack connections.";

uses connection;

}

container mid-percentile-c {

description

"Mid percentile of attack connections.";

uses connection;

}

container high-percentile-c {

description

"High percentile of attack connections.";

uses connection;

}

container peak-c {

description

"Peak attack connections.";

uses connection;

}

}

grouping connection-protocol-percentile {

description

"Total attack connections.";

list low-percentile-l {

key "protocol";

description

"Low percentile of attack connections.";

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.";

}

uses connection;

}

list mid-percentile-l {

key "protocol";

description

"Mid percentile of attack connections.";

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.";

}

uses connection;

}

list high-percentile-l {

key "protocol";

description

"Highg percentile of attack connections.";

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.";

}

uses connection;

}

list peak-l {

key "protocol";

description

"Peak attack connections.";

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.";

}

uses connection;

}

}

grouping attack-detail {

description

"Various information and details that describe the on-going

attacks that needs to be mitigated by the DOTS server.

The attack details need to cover well-known and common attacks

(such as a SYN Flood) along with new emerging or vendor-specific

attacks.";

leaf id {

type uint32;

description

"Vendor ID is a security vendor's Enterprise Number.";

}

leaf attack-id {

type string;

description

"Unique identifier assigned by the vendor for the attack.";

}

leaf attack-name {

type string;

description

"Textual representation of attack description. Natural Language

Processing techniques (e.g., word embedding) can possibly be used

to map the attack description to an attack type.";

}

leaf attack-severity {

type attack-severity;

description

"Severity level of an attack";

}

leaf start-time {

type uint64;

description

"The time the attack started. Start time is represented in seconds

relative to 1970-01-01T00:00:00Z in UTC time.";

}

leaf end-time {

type uint64;

description

"The time the attack ended. End time is represented in seconds

relative to 1970-01-01T00:00:00Z in UTC time.";

}

container source-count {

description

"Indicates the count of unique sources involved

in the attack.";

uses percentile;

}

}

grouping top-talker-aggregate {

description

"Top attack sources.";

list source-prefix {

key "source-prefix";

description

"IPv4 or IPv6 prefix identifying the attacker(s).";

leaf spoofed-status {

type boolean;

description

"Indicates whether this address is spoofed.";

}

leaf source-prefix {

type inet:ip-prefix;

description

"IPv4 or IPv6 prefix identifying the attacker(s).";

}

list total-attack-traffic {

key "unit";

description

"Total attack traffic issued from this source.";

uses traffic-unit;

}

container total-attack-connection {

description

"Total attack connections issued from this source.";

uses connection-percentile;

}

}

}

grouping top-talker {

description

"Top attack sources.";

list source-prefix {

key "source-prefix";

description

"IPv4 or IPv6 prefix identifying the attacker(s).";

leaf spoofed-status {

type boolean;

description

"Indicates whether this address is spoofed.";

}

leaf source-prefix {

type inet:ip-prefix;

description

"IPv4 or IPv6 prefix identifying the attacker(s).";

}

list total-attack-traffic {

key "unit";

description

"Total attack traffic issued from this source.";

uses traffic-unit;

}

container total-attack-connection {

description

"Total attack connections issued from this source.";

uses connection-protocol-percentile;

}

}

}

grouping baseline {

description

"Grouping for the telemetry baseline.";

uses ietf-data:target;

list total-traffic-normal-baseline {

key "unit protocol";

description

"Total traffic normal baselines.";

uses traffic-unit-protocol;

}

list total-connection-capacity {

key "protocol";

description

"Total connection capacity.";

leaf protocol {

type uint8;

description

"The transport protocol.

Values are taken from the IANA Protocol Numbers registry:

<https://www.iana.org/assignments/protocol-numbers/>.";

}

uses total-connection-capacity;

}

}

grouping pre-mitigation {

description

"Grouping for the telemetry data.";

list total-traffic {

key "unit protocol";

description

"Total traffic.";

uses traffic-unit-protocol;

}

list total-attack-traffic {

key "unit protocol";

description

"Total attack traffic per protocol.";

uses traffic-unit-protocol;

}

container total-attack-connection {

description

"Total attack connections.";

uses connection-protocol-percentile;

}

container attack-detail {

description

"Attack details.";

uses attack-detail;

container top-talker {

description

"Top attack sources.";

uses top-talker;

}

}

}

augment "/ietf-signal:dots-signal/ietf-signal:message-type/"

+ "ietf-signal:mitigation-scope/ietf-signal:scope" {

if-feature "dots-telemetry";

description

"Extends mitigation scope with telemetry update data.";

list total-traffic {

key "unit protocol";

description

"Total traffic.";

uses traffic-unit-protocol;

}

list total-attack-traffic {

key "unit";

description

"Total attack traffic.";

uses traffic-unit;

}

container total-attack-connection {

description

"Total attack connections.";

uses connection-percentile;

}

container attack-detail {

description

"Atatck details";

uses attack-detail;

container top-talker {

description

"Top attack sources.";

uses top-talker-aggregate;

}

}

}

augment "/ietf-signal:dots-signal/ietf-signal:message-type" {

if-feature "dots-telemetry";

description

"Add a new choice to enclose telemetry data in DOTS

signal channel.";

case telemetry-setup {

description

"Indicates the message is about telemetry.";

list telemetry {

key "cuid tsid";

description

"The telemetry data per DOTS client.";

leaf cuid {

type string;

description

"A unique identifier that is

generated by a DOTS client to prevent

request collisions. It is expected that the

cuid will remain consistent throughout the

lifetime of the DOTS client.";

}

leaf cdid {

type string;

description

"The cdid should be included by a server-domain

DOTS gateway to propagate the client domain

identification information from the

gateway's client-facing-side to the gateway's

server-facing-side, and from the gateway's

server-facing-side to the DOTS server.

It may be used by the final DOTS server

for policy enforcement purposes.";

}

leaf tsid {

type uint32;

description

"An identifier for the DOTS telemetry setup

data.";

}

choice setup-type {

description

"Can be a mitigation configuration, a pipe capacity,

or baseline message.";

case telemetry-config {

description

"Uses to set low, mid, and high percentile values.";

container current-config {

description

"Current configuration values.";

uses percentile-config;

uses unit-config;

leaf server-initiated-telemetry {

type boolean;

description

"Used by a DOTS client to enable/disable whether it

accepts pre-mitigation telemetry from the DOTS

server.";

}

}

container max-config-values {

description

"Maximum acceptable configuration values.";

config false;

uses percentile-config;

// Check if this is right place for indciating this capability

leaf server-initiated-telemetry {

type boolean;

description

"Indicates whether the DOTS server can be instructed

to send pre-mitigation telemetry. If set to FALSE

or the attribute is not present, this is an indication

that the server does not support this capability.";

}

}

container min-config-values {

description

"Minimum acceptable configuration values.";

config false;

uses percentile-config;

}

container supported-units {

description

"Supported units and default activation status.";

config false;

uses unit-config;

}

}

case pipe {

description

"Total pipe capacity of a DOTS client domain";

list total-pipe-capacity {

key "link-id unit";

description

"Total pipe capacity of a DOTS client domain.";

leaf link-id {

type nt:link-id;

description

"Identifier of an interconnection link.";

}

leaf capacity {

type uint64;

mandatory true;

description

"Pipe capacity.";

}

leaf unit {

type unit;

description

"The traffic can be measured in packets per

second (PPS) or kilo packets per second (Kpps) and Bits per

Second (BPS), and kilobytes per second or megabytes per second

or gigabytes per second.";

}

}

}

case baseline {

description

"Traffic baseline information";

list baseline {

key "id";

description

"Traffic baseline information";

leaf id {

type uint32;

must '. >= 1';

description

"A baseline entry identifier.";

}

uses baseline;

}

}

}

}

}

case telemetry {

description

"Indicates the message is about telemetry.";

list pre-mitigation {

key "cuid id";

description

"Pre-mitigation telemetry per DOTS client.";

leaf cuid {

type string;

description

"A unique identifier that is

generated by a DOTS client to prevent

request collisions. It is expected that the

cuid will remain consistent throughout the

lifetime of the DOTS client.";

}

leaf cdid {

type string;

description

"The cdid should be included by a server-domain

DOTS gateway to propagate the client domain

identification information from the

gateway's client-facing-side to the gateway's

server-facing-side, and from the gateway's

server-facing-side to the DOTS server.

It may be used by the final DOTS server

for policy enforcement purposes.";

}

leaf id {

type uint32;

description

"An identifier to uniquely demux telemetry data send using

the same message.";

}

container target {

description

"Indicates the target.";

uses ietf-data:target;

}

uses pre-mitigation;

}

}

}

}

<CODE ENDS>

10. YANG/JSON Mapping Parameters to CBOR

All DOTS telemetry parameters in the payload of the DOTS signal

channel MUST be mapped to CBOR types as shown in the following table:

+----------------------+-------------+------+---------------+--------+

| Parameter Name | YANG | CBOR | CBOR Major | JSON |

| | Type | Key | Type & | Type |

| | | | Information | |

+----------------------+-------------+------+---------------+--------+

| ietf-dots-signal-cha | | | | |

| nnel:telemetry | container |32776 | 5 map | Object |

| tsid | uint32 |32777 | 0 unsigned | Number |

| telemetry-config | container |32778 | 5 map | Object |

| low-percentile | decimal64 |32779 | 6 tag 4 | |

| | | | [-2, integer]| String |

| mid-percentile | decimal64 |32780 | 6 tag 4 | |

| | | | [-2, integer]| String |

| high-percentile | decimal64 |32781 | 6 tag 4 | |

| | | | [-2, integer]| String |

| unit-config | list |32782 | 4 array | Array |

| unit | enumeration |32783 | 0 unsigned | String |

| unit-status | boolean |32784 | 7 bits 20 | False |

| | | | 7 bits 21 | True |

| total-pipe-capability| list |32785 | 4 array | Array |

| pipe | uint64 |32786 | 0 unsigned | String |

| pre-mitigation | list |32787 | 4 array | Array |

| ietf-dots-signal-cha | | | | |

| nnel:telemetry-setup | container |32788 | 5 map | Object |

| total-traffic- | | | | |

| normal-baseline | list |32789 | 4 array | Array |

| low-percentile-g | yang:gauge64|32790 | 0 unsigned | String |

| mid-percentile-g | yang:gauge64|32791 | 0 unsigned | String |

| high-percentile-g | yang:gauge64|32792 | 0 unsigned | String |

| peak-g | yang:gauge64|32793 | 0 unsigned | String |

| total-attack-traffic | list |32794 | 4 array | Array |

| total-traffic | list |32795 | 4 array | Array |

| total-connection- | | | | |

| capacity | list |32796 | 4 array | Array |

| connection | uint64 |32797 | 0 unsigned | String |

| connection-client | uint64 |32798 | 0 unsigned | String |

| embryonic | uint64 |32799 | 0 unsigned | String |

| embryonic-client | uint64 |32800 | 0 unsigned | String |

| connection-ps | uint64 |32801 | 0 unsigned | String |

| connection-client-ps | uint64 |32802 | 0 unsigned | String |

| request-ps | uint64 |32803 | 0 unsigned | String |

| request-client-ps | uint64 |32804 | 0 unsigned | String |

| partial-request-ps | uint64 |32805 | 0 unsigned | String |

| partial-request- | | | | |

| client-ps | uint64 |32806 | 0 unsigned | String |

| total-attack- | | | | |

| connection | container |32807 | 5 map | Object |

| low-percentile-l | list |32808 | 4 array | Array |

| mid-percentile-l | list |32809 | 4 array | Array |

| high-percentile-l | list |32810 | 4 array | Array |

| peak-l | list |32811 | 4 array | Array |

| attack-detail | container |32812 | 5 map | Object |

| id | uint32 |32813 | 0 unsigned | Number |

| attack-id | string |32814 | 3 text string | String |

| attack-name | string |32815 | 3 text string | String |

| attack-severity | enumeration |32816 | 0 unsigned | String |

| start-time | uint64 |32817 | 0 unsigned | String |

| end-time | uint64 |32819 | 0 unsigned | String |

| source-count | container |32820 | 5 map | Object |

| top-talker | container |32821 | 5 map | Object |

| spoofed-status | boolean |32822 | 7 bits 20 | False |

| | | | 7 bits 21 | True |

| low-percentile-c | container |32823 | 5 map | Object |

| mid-percentile-c | container |32824 | 5 map | Object |

| high-percentile-c | container |32825 | 5 map | Object |

| peak-c | container |32826 | 5 map | Object |

| baseline | container |32827 | 5 map | Object |

| current-config | container |32828 | 5 map | Object |

| max-config-values | container |32829 | 5 map | Object |

| min-config-values | container |32830 | 5 map | Object |

| supported-units | container |32831 | 5 map | Object |

| server-initiated- | boolean |32832 | 7 bits 20 | False |

| telemetry | | | 7 bits 21 | True |

+----------------------+-------------+------+---------------+--------+

11. IANA Considerations

11.1. DOTS Signal Channel CBOR Key Values

This specification registers the DOTS telemetry attributes in the

IANA "DOTS Signal Channel CBOR Key Values" registry available at

https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-

cbor-key-values.

The DOTS telemetry attributes defined in this specification are

comprehension-optional parameters.

o Note to the RFC Editor: (1) CBOR keys are assigned from the

32768-49151 range. (2) Please assign the following suggested

values.

+----------------------+-------+-------+------------+---------------+

| Parameter Name | CBOR | CBOR | Change | Specification |

| | Key | Major | Controller | Document(s) |

| | Value | Type | | |

+----------------------+-------+-------+------------+---------------+

| ietf-dots-signal-cha | 32776 | 5 | IESG | [RFCXXXX] |

| nnel:telemetry | | | | |

| tsid | 32777 | 0 | IESG | [RFCXXXX] |

| telemetry-config | 32778 | 5 | IESG | [RFCXXXX] |

| low-percentile | 32779 | 6tag4 | IESG | [RFCXXXX] |

| mid-percentile | 32780 | 6tag4 | IESG | [RFCXXXX] |

| high-percentile | 32781 | 6tag4 | IESG | [RFCXXXX] |

| unit-config | 32782 | 4 | IESG | [RFCXXXX] |

| unit | 32783 | 0 | IESG | [RFCXXXX] |

| unit-status | 32784 | 7 | IESG | [RFCXXXX] |

| total-pipe-capability| 32785 | 4 | IESG | [RFCXXXX] |

| pipe | 32786 | 0 | IESG | [RFCXXXX] |

| pre-mitigation | 32787 | 4 | IESG | [RFCXXXX] |

| ietf-dots-signal-cha | 32788 | 5 | IESG | [RFCXXXX] |

| nnel:telemetry | | | | |

| total-traffic- | 32789 | 4 | IESG | [RFCXXXX] |

| normal-baseline | | | | |

| low-percentile-g | 32790 | 0 | IESG | [RFCXXXX] |

| mid-percentile-g | 32791 | 0 | IESG | [RFCXXXX] |

| high-percentile-g | 32792 | 0 | IESG | [RFCXXXX] |

| peak-g | 32793 | 0 | IESG | [RFCXXXX] |

| total-attack-traffic | 32794 | 4 | IESG | [RFCXXXX] |

| total-traffic | 32795 | 4 | IESG | [RFCXXXX] |

| total-connection- | 32796 | 4 | IESG | [RFCXXXX] |

| capacity | | | | |

| connection | 32797 | 0 | IESG | [RFCXXXX] |

| connection-client | 32798 | 0 | IESG | [RFCXXXX] |

| embryonic | 32799 | 0 | IESG | [RFCXXXX] |

| embryonic-client | 32800 | 0 | IESG | [RFCXXXX] |

| connection-ps | 32801 | 0 | IESG | [RFCXXXX] |

| connection-client-ps | 32802 | 0 | IESG | [RFCXXXX] |

| request-ps | 32803 | 0 | IESG | [RFCXXXX] |

| request-client-ps | 32804 | 0 | IESG | [RFCXXXX] |

| partial-request-ps | 32805 | 0 | IESG | [RFCXXXX] |

| partial-request- | 32806 | 0 | IESG | [RFCXXXX] |

| client-ps | | | | |

| total-attack- | 32807 | 5 | IESG | [RFCXXXX] |

| connection | | | | |

| low-percentile-l | 32808 | 4 | IESG | [RFCXXXX] |

| mid-percentile-l | 32809 | 4 | IESG | [RFCXXXX] |

| high-percentile-l | 32810 | 4 | IESG | [RFCXXXX] |

| peak-l | 32811 | 4 | IESG | [RFCXXXX] |

| attack-detail | 32812 | 5 | IESG | [RFCXXXX] |

| id | 32813 | 0 | IESG | [RFCXXXX] |

| attack-id | 32814 | 3 | IESG | [RFCXXXX] |

| attack-name | 32815 | 3 | IESG | [RFCXXXX] |

| attack-severity | 32816 | 0 | IESG | [RFCXXXX] |

| start-time | 32817 | 0 | IESG | [RFCXXXX] |

| end-time | 32819 | 0 | IESG | [RFCXXXX] |

| source-count | 32820 | 5 | IESG | [RFCXXXX] |

| top-talker | 32821 | 5 | IESG | [RFCXXXX] |

| spoofed-status | 32822 | 7 | IESG | [RFCXXXX] |

| low-percentile-c | 32823 | 5 | IESG | [RFCXXXX] |

| mid-percentile-c | 32824 | 5 | IESG | [RFCXXXX] |

| high-percentile-c | 32825 | 5 | IESG | [RFCXXXX] |

| peak-c | 32826 | 5 | IESG | [RFCXXXX] |

| ietf-dots-signal-cha | 32827 | 5 | IESG | [RFCXXXX] |

| current-config | 32828 | 5 | IESG | [RFCXXXX] |

| max-config-value | 32829 | 5 | IESG | [RFCXXXX] |

| min-config-values | 32830 | 5 | IESG | [RFCXXXX] |

| supported-units | 32831 | 5 | IESG | [RFCXXXX] |

| server-initiated- | 32832 | 7 | IESG | [RFCXXXX] |

| telemetry | | | | |

+----------------------+-------+-------+------------+---------------+

11.2. DOTS Signal Channel Conflict Cause Codes

This specification requests IANA to assign a new code from the "DOTS

Signal Channel Conflict Cause Codes" registry available at

https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-

conflict-cause-codes.

Code Label Description Reference

TBA overlapping-pipes Overlapping pipe scope [RFCXXXX]

11.3. DOTS Signal Telemetry YANG Module

This document requests IANA to register the following URI in the "ns"

subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in

the "YANG Module Names" subregistry [RFC7950] within the "YANG

Parameters" registry.

name: ietf-dots-telemetry

namespace: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry

maintained by IANA: N

prefix: dots-telemetry

reference: RFC XXXX

12. Security Considerations

Security considerations in [I-D.ietf-dots-signal-channel] need to be

taken into consideration.

13. Contributors

The following individuals have contributed to this document:

o Li Su, CMCC, Email: suli@chinamobile.com

o Jin Peng, CMCC, Email: pengjin@chinamobile.com

o Pan Wei, Huawei, Email: william.panwei@huawei.com

14. Acknowledgements

The authors would like to thank Flemming Andreasen, Liang Xia, and

Kaname Nishizuka co-authors of https://tools.ietf.org/html/draft-

doron-dots-telemetry-00 draft and everyone who had contributed to

that document.

Authors would like to thank Kaname Nishizuka, Jon Shallow, Wei Pan

and Yuuhei Hayashi for comments and review.

15. References

15.1. Normative References

[Enterprise-Numbers]

"Private Enterprise Numbers", 2005, <http://www.iana.org/

assignments/enterprise-numbers.html>.

[I-D.ietf-dots-data-channel]

Boucadair, M. and T. Reddy.K, "Distributed Denial-of-

Service Open Threat Signaling (DOTS) Data Channel

Specification", draft-ietf-dots-data-channel-31 (work in

progress), July 2019.

[I-D.ietf-dots-signal-call-home]

Reddy.K, T., Boucadair, M., and J. Shallow, "Distributed

Denial-of-Service Open Threat Signaling (DOTS) Signal

Channel Call Home", draft-ietf-dots-signal-call-home-07

(work in progress), November 2019.

[I-D.ietf-dots-signal-channel]

Reddy.K, 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-41 (work in progress), January

2020.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",

RFC 6991, DOI 10.17487/RFC6991, July 2013,

<https://www.rfc-editor.org/info/rfc6991>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object

Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,

October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained

Application Protocol (CoAP)", RFC 7641,

DOI 10.17487/RFC7641, September 2015,

<https://www.rfc-editor.org/info/rfc7641>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",

RFC 7950, DOI 10.17487/RFC7950, August 2016,

<https://www.rfc-editor.org/info/rfc7950>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in

the Constrained Application Protocol (CoAP)", RFC 7959,

DOI 10.17487/RFC7959, August 2016,

<https://www.rfc-editor.org/info/rfc7959>.

[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>.

15.2. Informative References

[I-D.ietf-dots-signal-filter-control]

Nishizuka, K., Boucadair, M., Reddy.K, T., and T. Nagata,

"Controlling Filtering Rules Using Distributed Denial-of-

Service Open Threat Signaling (DOTS) Signal Channel",

draft-ietf-dots-signal-filter-control-02 (work in

progress), September 2019.

[I-D.ietf-dots-use-cases]

Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia,

L., and K. Nishizuka, "Use cases for DDoS Open Threat

Signaling", draft-ietf-dots-use-cases-20 (work in

progress), September 2019.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis,

"Framework for IP Performance Metrics", RFC 2330,

DOI 10.17487/RFC2330, May 1998,

<https://www.rfc-editor.org/info/rfc2330>.

[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>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N.,

Ananthakrishnan, H., and X. Liu, "A YANG Data Model for

Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March

2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open

Threat Signaling (DOTS) Requirements", RFC 8612,

DOI 10.17487/RFC8612, May 2019,

<https://www.rfc-editor.org/info/rfc8612>.

Authors' Addresses

Mohamed Boucadair (editor)

Orange

Rennes 35000

France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy (editor)

McAfee, Inc.

Embassy Golf Link Business Park

Bangalore, Karnataka 560071

India

Email: kondtir@gmail.com

Ehud Doron

Radware Ltd.

Raoul Wallenberg Street

Tel-Aviv 69710

Israel

Email: ehudd@radware.com

Meiling Chen

CMCC

32, Xuanwumen West

BeiJing, BeiJing 100053

China

Email: chenmeiling@chinamobile.com