DOTS
Internet-Draft
Intended status: Standards Track
Expires: April 6, 2020

T. Reddy
McAfee
M. Boucadair
Orange
E. Doron
Radware Ltd.
M. Chen
CMCC

October 04, 2019

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry draft-reddy-dots-telemetry-03

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.

Table of Contents

1. Introduction

The Internet security 'battle' between the adversary and security countermeasures is an everlasting one. 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. 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, DNS, SNMP, SSDP, and so on. Application layer attacks target various applications. Typical examples include attacks against HTTP/HTTPS, DNS, SIP, SMTP, and so on. 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 merciless 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 Distributed Denial of Service (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 are 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 & 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 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 protect 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, for cases of 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 current level of traffic client can observe from server. The server should activate other mechanisms to ensure it does not saturate the client's pipes unintentionally. The rate-limit action defined in [I-D.ietf-dots-data-channel] can be a reasonable candidate to achieve this objective; the client can ask for the type of traffic (such as ICMP, UDP, TCP port 80) it prefers to limit.

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. DOTS Telemetry Attributes

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 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 were 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.

4.1. Pre-mitigation DOTS Telemetry Attributes

The pre-mitigation telemetry attributes are indicated by the path-suffix '/telemetry'. The '/telemetry' 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 the DOTS client to the DOTS server.

- 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 a DOTS client, e.g., Indicate a favorite measurement unit? Indicate a minimum notification interval?

4.1.1. Total Traffic Normal Baseline

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.

4.1.2. Total Pipe Capability

The limit of traffic volume, 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. These attributes represents the DOTS client domain pipe limit.

- NOTE: Multi-homing case to be considered.

4.1.3. 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.

4.1.4. 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.

4.1.5. Total Connections Capacity

If the target is subjected to resource consuming DDoS attack, 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 server. The threshold is transport-protocol specific because the target server could support multiple protocols.
- The maximum number of simultaneous connections that are allowed to the target server per client.
- 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.
- The maximum number of simultaneous embryonic connections that are allowed to the target server per client.
- The maximum number of connections allowed per second to the target server.
- The maximum number of connections allowed per second to the target server per client.
- The maximum number of requests allowed per second to the target server.
- The maximum number of requests allowed per second to the target server per client.
- The maximum number of partial requests allowed per second to the target server.
- The maximum number of partial requests allowed per second to the target server per client.

4.1.6. 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:

- The number of simultaneous attack connections to the target server.
- The number of simultaneous embryonic connections to the target server.
- The number of attack connections per second to the target server.
- The number of attack requests to the target server.

4.1.7. Attack Details

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. The attack details can also be signaled from the DOTS server to the DOTS client. 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 client can use the attack details to decide whether to trigger the mitigation request or not. Further, 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. The DOTS client can receive asynchronous notifications of the attack details from the DOTS server using the Observe option defined in [RFC7641].

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 (0), critical (1) and alert (2).

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 4.1.6 are applicable for representing the attack.

This is an optional sub-attribute.

- List of top talkers targeting the victim. 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 4.1.6 are applicable here for representing the attack per talker. This is an optional sub-attribute.

4.2. 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.

4.2.1. 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.

4.2.2. Attack Details

The overall attack details as observed from the DOTS client perspective during the active mitigation service. The same attributes defined in Section 4.1.7 are applicable here.

4.3. 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.

4.3.1. 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 4.1.7 are applicable here for describing the attacks detected and mitigated.

5. DOTS Telemetry Configuration

5.1. 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 percentiles values used as baseline for telemetry data. In reference to the example shown in Figure 1, the DOTS client modifies all percentile reference values.

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

{
  "ietf-dots-telemetry:telemetry-config": {
    "low-percentile": 5.00,
    "mid-percentile": 65.00,
    "high-percentile": 95.00
  }
}
```

Figure 1: PUT to Convey the DOTS Telemetry Configuration

The following additional Uri-Path parameter is defined:

tcid: Telemetry Configuration Identifier is an identifier for the DOTS telemetry configuration data represented as an integer. This identifier MUST be generated by DOTS clients. 'tcid' values MUST increase monotonically (when a new PUT is generated by a DOTS client to convey the configuration parameters for the telemetry).

This is a mandatory attribute.

At least one configurable attribute MUST be present in the PUT request.

The PUT request with a higher numeric 'tcid' value overrides the DOTS telemetry configuration data installed by a PUT request with a lower numeric 'tcid' value. To avoid maintaining a long list of 'tcid' requests from a DOTS client, the lower numeric 'tcid' MUST be automatically deleted and no longer available at the DOTS server.

The DOTS server indicates the result of processing the PUT request using CoAP response codes:

- If the request is missing a mandatory attribute, does not include a 'tcid' Uri-Path, or contains one or more invalid or unknown parameters, 4.00 (Bad Request) MUST be returned in the response.

- If the DOTS server does not find the 'tcid' 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.

- If the DOTS server finds the 'tcid' 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.

- If any of the enclosed configurable attribute values are not acceptable to the DOTS server, 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.

A DOTS client may issue a GET message with 'tcid' Uri-Path parameter to retrieve the negotiated configuration. The response does not need to include 'tcid' in its message body.

5.2. Delete DOTS Telemetry Configuration

A DELETE request is used to delete the installed DOTS telemetry configuration data (Figure 2).

```
  Header: DELETE (Code=0.04)
  Uri-Path: ".well-known"
  Uri-Path: "dots"
  Uri-Path: "telemetry"
  Uri-Path: "tcid=123"
```

Figure 2: Delete Telemetry Configuration

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.

Upon bootstrapping or reboot, a DOTS client MAY send a DELETE request to set the telemetry parameters to default values. Such a request does not include any 'tcid'.

6. DOTS Telemetry YANG Module

6.1. Tree Structure

This document defines the YANG module "ietf-dots-telemetry", which has the following tree structure. It augments the "ietf-dots-signal" with a new message type called "telemetry" and the "mitigation-scope" type message with telemetry data.

Notes: (1) Check naming conflict to ease CBOR mapping (e.g, low-percentile is defined as yang:gauge64, list, or container). Distinct names may be considered. (2) "protocol" is not indicated in the telemetry data of "mitigation-scope" message type because the mitigation request may include a "protocol". Similarly, "target-*" is not included in the in the telemetry data of "mitigation-scope" message type because the mitigation request must include at least one of the "target-*" attribute.

```
module: ietf-dots-telemetry
  augment /ietf-signal:dots-signal/ietf-signal:message-type
          /ietf-signal:mitigation-scope/ietf-signal:scope:
    +--rw total-attack-traffic* [unit] {dots-telemetry}?
    |  +--rw unit               unit
    |  +--rw low-percentile?    yang:gauge64
    |  +--rw mid-percentile?    yang:gauge64
    |  +--rw high-percentile?   yang:gauge64
    |  +--rw peak?              yang:gauge64
    +--rw total-attack-connection {dots-telemetry}?
    |  +--rw low-percentile
    |  |  +--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
|  |  +--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
|  |  +--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
|     +--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 vendor-id?         uint32
   +--rw attack-id?         string
   +--rw attack-name?       string
   +--rw attack-severity?   attack-severity
   +--rw start-time?        uint64
   +--rw end-time?          uint64
   +--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?         yang:gauge64
         |  +--rw mid-percentile?         yang:gauge64
         |  +--rw high-percentile?        yang:gauge64
         |  +--rw peak?                   yang:gauge64
         +--rw total-attack-connection
            +--rw low-percentile
            |  +--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
            |  +--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
                  |  +--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
                     +--rw connection?           yang:gauge64
                     +--rw embryonic?            yang:gauge64
                     +--rw connection-ps?        yang:gauge64
                     +--rw request-ps?           yang:gauge64
                     +--rw partial-request-ps?   yang:gauge64
    augment /ietf-signal:dots-signal/ietf-signal:message-type:
      +--:(telemetry) {dots-telemetry}?
         +--rw telemetry-config
         |  +--rw tcid               uint32
         |  +--rw low-percentile?    percentile
         |  +--rw mid-percentile?    percentile
         |  +--rw high-percentile?   percentile
         +--rw total-pipe-capability* [unit]
         |  +--rw unit    unit
         |  +--rw pipe?   uint64
         +--rw pre-mitigation* [telemetry-id]
            +--rw telemetry-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-normal-baseline* [unit protocol]
            |  +--rw unit               unit
            |  +--rw protocol           uint8
            |  +--rw low-percentile?    yang:gauge64
            |  +--rw mid-percentile?    yang:gauge64
            |  +--rw high-percentile?   yang:gauge64
            |  +--rw peak?              yang:gauge64
            +--ro total-attack-traffic* [unit protocol]
            |  +--ro unit               unit
            |  +--ro protocol           uint8
            |  +--ro low-percentile?    yang:gauge64
            |  +--ro mid-percentile?    yang:gauge64
            |  +--ro high-percentile?   yang:gauge64
            |  +--ro peak?              yang:gauge64
```

```
        +--ro total-traffic* [unit protocol]
        |  +--ro unit               unit
        |  +--ro protocol           uint8
        |  +--ro low-percentile?    yang:gauge64
        |  +--ro mid-percentile?    yang:gauge64
        |  +--ro high-percentile?   yang:gauge64
        |  +--ro peak?              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
        +--ro total-attack-connection
        |  +--ro low-percentile* [protocol]
        |  |  +--ro protocol             uint8
        |  |  +--ro connection?          yang:gauge64
        |  |  +--ro embryonic?           yang:gauge64
        |  |  +--ro connection-ps?       yang:gauge64
        |  |  +--ro request-ps?          yang:gauge64
        |  |  +--ro partial-request-ps?  yang:gauge64
        |  +--ro mid-percentile* [protocol]
        |  |  +--ro protocol             uint8
        |  |  +--ro connection?          yang:gauge64
        |  |  +--ro embryonic?           yang:gauge64
        |  |  +--ro connection-ps?       yang:gauge64
        |  |  +--ro request-ps?          yang:gauge64
        |  |  +--ro partial-request-ps?  yang:gauge64
        |  +--ro high-percentile* [protocol]
        |  |  +--ro protocol             uint8
        |  |  +--ro connection?          yang:gauge64
        |  |  +--ro embryonic?           yang:gauge64
        |  |  +--ro connection-ps?       yang:gauge64
        |  |  +--ro request-ps?          yang:gauge64
        |  |  +--ro partial-request-ps?  yang:gauge64
        |  +--ro peak* [protocol]
        |     +--ro protocol             uint8
        |     +--ro connection?          yang:gauge64
        |     +--ro embryonic?           yang:gauge64
        |     +--ro connection-ps?       yang:gauge64
        |     +--ro request-ps?          yang:gauge64
        |     +--ro partial-request-ps?  yang:gauge64
```

```
          +--ro attack-detail
             +--ro vendor-id?         uint32
             +--ro attack-id?         string
             +--ro attack-name?       string
             +--ro attack-severity?   attack-severity
             +--ro start-time?        uint64
             +--ro end-time?          uint64
             +--ro top-talker
                +--ro source-prefix* [source-prefix]
                   +--ro spoofed-status?           boolean
                   +--ro source-prefix             inet:ip-prefix
                   +--ro total-attack-traffic* [unit]
                   |  +--ro unit                   unit
                   |  +--ro low-percentile?        yang:gauge64
                   |  +--ro mid-percentile?        yang:gauge64
                   |  +--ro high-percentile?       yang:gauge64
                   |  +--ro peak?                  yang:gauge64
                   +--ro total-attack-connection
                      +--ro low-percentile* [protocol]
                      |  +--ro protocol              uint8
                      |  +--ro connection?           yang:gauge64
                      |  +--ro embryonic?            yang:gauge64
                      |  +--ro connection-ps?        yang:gauge64
                      |  +--ro request-ps?           yang:gauge64
                      |  +--ro partial-request-ps?   yang:gauge64
                      +--ro mid-percentile* [protocol]
                      |  +--ro protocol              uint8
                      |  +--ro connection?           yang:gauge64
                      |  +--ro embryonic?            yang:gauge64
                      |  +--ro connection-ps?        yang:gauge64
                      |  +--ro request-ps?           yang:gauge64
                      |  +--ro partial-request-ps?   yang:gauge64
                      +--ro high-percentile* [protocol]
                      |  +--ro protocol              uint8
                      |  +--ro connection?           yang:gauge64
                      |  +--ro embryonic?            yang:gauge64
                      |  +--ro connection-ps?        yang:gauge64
                      |  +--ro request-ps?           yang:gauge64
                      |  +--ro partial-request-ps?   yang:gauge64
                      +--ro peak* [protocol]
                         +--ro protocol              uint8
                         +--ro connection?           yang:gauge64
                         +--ro embryonic?            yang:gauge64
                         +--ro connection-ps?        yang:gauge64
                         +--ro request-ps?           yang:gauge64
                         +--ro partial-request-ps?   yang:gauge64
```

6.2. YANG Module

This module uses types defined in [RFC6991].

```
<CODE BEGINS> file "ietf-dots-telemetry@2019-10-01.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";
  }

  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) 2019 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 2019-10-01 {
    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 atatck 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 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 low-percentile {
      type percentile;
      default "10.00";
      description
        "Low percentile.";
    }
    leaf mid-percentile {
      type percentile;
      default "50.00";
      description
        "Mid percentile.";
    }
    leaf high-percentile {
      type percentile;
      default "90.00";
      description
        "High percentile.";
    }
  }

  grouping traffic {
    description
      "Generic grouping for traffic percentile.";
    leaf low-percentile {
      type yang:gauge64;
      description
        "Low traffic percentile.";
    }
    leaf mid-percentile {
      type yang:gauge64;
      description
        "Mid traffic percentile.";
    }
    leaf high-percentile {
      type yang:gauge64;
      description
        "High traffic percentile.";
    }
    leaf peak {
      type yang:gauge64;
      description
        "Peak";
    }
  }

  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 traffic;
  }

  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 traffic;
  }

  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 conenctions 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 {
      description
        "Low percentile of attack connections.";
      uses connection;
    }
    container mid-percentile {
      description
        "Mid percentile of attack connections.";
      uses connection;
    }
    container high-percentile {
      description
        "Highg percentile of attack connections.";
      uses connection;
    }
    container peak {
      description
        "Peak attack connections.";
      uses connection;
    }
  }

  grouping connection-protocol-percentile {
    description
      "Total attack connections.";
    list low-percentile {
      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 {
      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 {
      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 {
      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 vendor-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.";
    }
    //uses ietf-data:target;
  }

  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;
      }
    }
    /*list source-port-range {
      key "lower-port";
      description
        "Port range. When only lower-port is
         present, it represents a single port number.";
      leaf lower-port {
        type inet:port-number;
        mandatory true;
        description
          "Lower port number of the port range.";
      }
      leaf upper-port {
        type inet:port-number;
        must '. >= ../lower-port' {
          error-message
            "The upper port number must be greater than
             or equal to lower port number.";
        }
        description
          "Upper port number of the port range.";
      }
    }
    list source-icmp-type-range {
      key "lower-type";
      description
        "ICMP type range. When only lower-type is
         present, it represents a single ICMP type.";
      leaf lower-type {
        type uint8;
        mandatory true;
        description
          "Lower ICMP type of the ICMP type range.";
      }
      leaf upper-type {
        type uint8;
        must '. >= ../lower-type' {
          error-message
```

```
            "The upper ICMP type must be greater than
             or equal to lower ICMP type.";
        }
        description
          "Upper type of the ICMP type range.";
      }
    }*/
  }

  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 pre-mitigation {
    description
      "Grouping for the telemetry data.";
    list total-traffic-normal-baseline {
      key "unit protocol";
      description
        "Total traffic normal baselines.";
      uses traffic-unit-protocol;
```

```
    }
    list total-attack-traffic {
      key "unit protocol";
      config false;
      description
        "Total attack traffic per protocol.";
      uses traffic-unit-protocol;
    }
    list total-traffic {
      key "unit protocol";
      config false;
      description
        "Total traffic.";
      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;
    }
    container total-attack-connection {
      description
        "Total attack connections.";
      config false;
      uses connection-protocol-percentile;
    }
    container attack-detail {
      description
        "Attack details.";
      config false;
      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-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 {
      description
        "Indicates the message is about telemetry.";
      container telemetry-config {
        description
          "Uses to set low, mid, and high percentile values.";
        leaf tcid {
          type uint32;
          mandatory true;
          description
            "An identifier for the DOTS telemetry
             configuration data.";
        }
        uses percentile-config;
      }
      list total-pipe-capability {
        key "unit";
```

```
        description
          "Total pipe capacity of a DOTS client domain.";
        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 pipe {
          type uint64;
          description
            "Mid traffic percentile.";
        }
      }
      list pre-mitigation {
        key "telemetry-id";
        description
          "Pre-mitigation telemetry.";
        leaf telemetry-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>
```

7. IANA Considerations

7.1. DOTS Signal Channel CBOR Mappings Registry

This specification registers the DOTS telemetry attributes in the IANA "DOTS Signal Channel CBOR Mappings" registry established by [I-D.ietf-dots-signal-channel].

The DOTS telemetry attributes defined in this specification are comprehension-optional parameters.

- Note to the RFC Editor: Please delete (TBD1)-(TBD5) once CBOR keys are assigned from the 0x8000 - 0xBFFF range.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
TODO				

7.2. 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
```

8. Security Considerations

Security considerations in [I-D.ietf-dots-signal-channel] need to be taken into consideration.

9. Contributors

The following individuals have contributed to this document:

- Li Su, CMCC, Email: suli@chinamobile.com

- Jin Peng, CMCC, Email: pengjin@chinamobile.com

10. 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.

11. References

11.1. 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 R. 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]
K, R., Boucadair, M., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home", draft-ietf-dots-signal-call-home-06 (work in progress), September 2019.

[I-D.ietf-dots-signal-channel]
K, R., 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-37 (work in progress), July 2019.

[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>.

[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>.

11.2. Informative References

[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.

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

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

Authors' Addresses

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

Email: kondtir@gmail.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

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