

DOTS
Internet-Draft
Intended status: Informational
Expires: December 26, 2020

M. Chen
Li. Su
China Mobile
June 24, 2020

A method for dots server deployment
draft-chen-dots-server-hierarchical-deployment-03

Abstract

As DOTS is used for DDoS Mitigation signaling, there are different deployment scenarios for DOTS agents deployment depending on the network topology. This document made recommendations for DOTS Server deployment, include ISP and enterprise deployment scenarios. The goal is to provide some guidance for DOTS agents deployment.

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 December 26, 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	2
2. Terminology	3
3. DOTS server Considerations	3
4. DOTS server deployment inside an ISP	4
4.1. DOTS Agents Deployment	4
4.2. DOTS Agents interfaces	7
4.2.1. Bandwidth consuming attack	8
4.2.2. Host resource consuming attack	8
5. DOTS server deployment between ISPs	8
6. DOTS server deployment for Enterprise	9
7. Security Considerations	10
8. IANA Considerations	10
9. Acknowledgement	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Authors' Addresses	12

1. Introduction

DDoS Open Threat Signaling (DOTS) is a protocol to standardize real-time signaling, threat-handling requests[I-D.ietf-dots-signal-channel], when attack target is under attack, dots client send mitigation request to dots server for help, If the mitigation request contains enough messages of the attack, then the mitigator can respond very effectively.

In the architecture draft[I-D.ietf-dots-architecture], when comes to the deployment topic, it says this does not necessarily imply that the attack target and the DOTS client have to be co-located in the same administrative domain, but it is expected to be a common scenario. Although co-location of DOTS server and mitigator within the same domain is expected to be a common deployment model, it is assumed that operators may require alternative models.

In the DOTS server discovery draft[I-D.ietf-dots-server-discovery], it is says that a key point in the deployment of DOTS is the ability of network operators to be able to configure DOTS clients with the correct DOTS server(s) information consistently.

In the DOTS multihoming draft[I-D.ietf-dots-multihoming], it provides deployment recommendations for DOTS client and DOTS gateway, it is says when conveying a mitigation request to protect the attack target, the DOTS client among the DOTS servers available Must select a DOTS server whose network has assigned the prefixes from which target prefixes and target IP addresses are derived. This implies

that if no appropriate DOTS server is found, the DOTS client must not send the mitigation request to any DOTS server. So in this document, we give some dots server deployment consideration as the title suggests we prefer hierarchical deployment.

This is DOTS server deployment guidance for operators, We've written about our experience as an ISP, and we hope that other scenarios will contribute as well.

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 [RFC2119]

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

The terminology related to YANG data modules is defined in [RFC7950]

In addition, this document uses the terms defined below:

dots svr: abbreviation of dots server.

ISP: Internet service provider.

Orchestrator: With the function of DOTS server that can receive messages from clients and made decisions for mitigators selection.

netflow/ipfix collector: Flow collector used for DDoS attack detection.

3. DOTS server Considerations

When take dots server deployment into consideration, one thing must be involved is mitigator that can provide DDoS mitigation service. So far, how many network devices can play the role of mitigator, we make a summarized list as follows:

- o Router.
- o Special cleaning equipment, such as flow clean device and clean center.
- o Network security equipment, such as firewall, IPS and WAF.
- o Servers that websites can hidden behind them.

There are several requirements for DOTS server deployment that may be , and is consistent with other drafts:

- o DOTS server and mitigator are in the same administrative domain.
- o DOTS server can go directly to the mitigator which had best go through without any other DOTS agents.
- o DOTS server has the permissions for scheduling on mitigators.
- o DOTS server has the ability to know the address of attack target belong to which mitigator, if DOTS server hasn't matched attack target to mitigators, DOTS server need to configure default mitigators.

4. DOTS server deployment inside an ISP

4.1. DOTS Agents Deployment

From the internal structure of ISP, the whole network can abstract as a three-level network logically. The hierarchy of the network can be adjusted according to the size of the network. In addition to having its own business, the upper network is responsible for connectivity between the lower networks. It's worth noting that there are usually Internet Data Centers(IDC), high bandwidth demand customers(such as online game companies) and VIP customer centers(such as financial clients) distributed in each level network, but most of these services are typically placed on a secondary network.

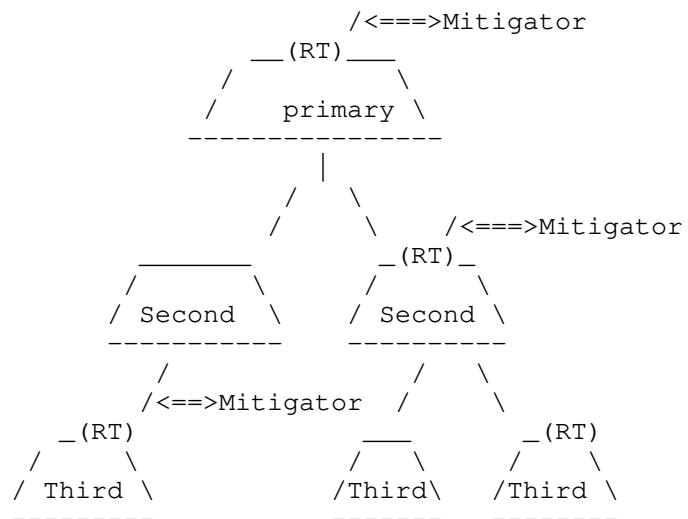


Figure 1: ISP multilevel network

There are mitigators such as cleaning centers in each regional network. Select the second level network for detailed description, the cleaning equipment is attached to the exit router, and Detector is concatenated on the link, usually detector could be one type of netflow/ipfix collector, sometimes could be firewall or IDS(Intrusion Detection System), they could able to find some types of DDoS attacks. Attacks from two different sources occur inside the Second network as follows:

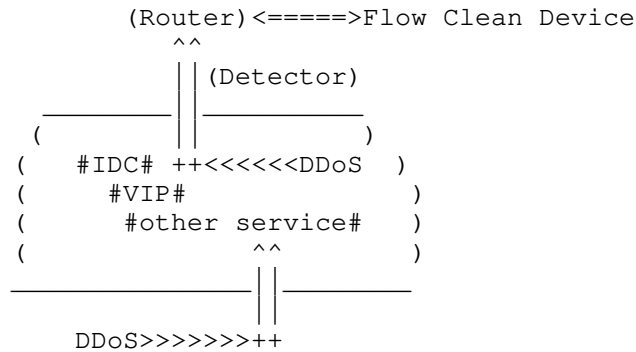


Figure 2:Two DDoS attack paths

There are only two attack source paths under this structure: One is an attack launched within the network, flowing to the upper level of the network. The other is that low-level networks launch attacks and flow to the upper level of networks, and pass through the intermediate level network. Internal DDoS attacks is out of scope in this draft.

In this case, DOTS clients might consider to be deployed internally. When DDoS attack occurs, attack target inside the Second network may sense being attacked, such as customer complaints and the processing speed is slower than before. attack target then inform Detector(DOTS client) making mitigation request to DOTS server(Router), and the traffic mitigation is then triggered. DOTS server and Mitigator are in the same administrative domain.

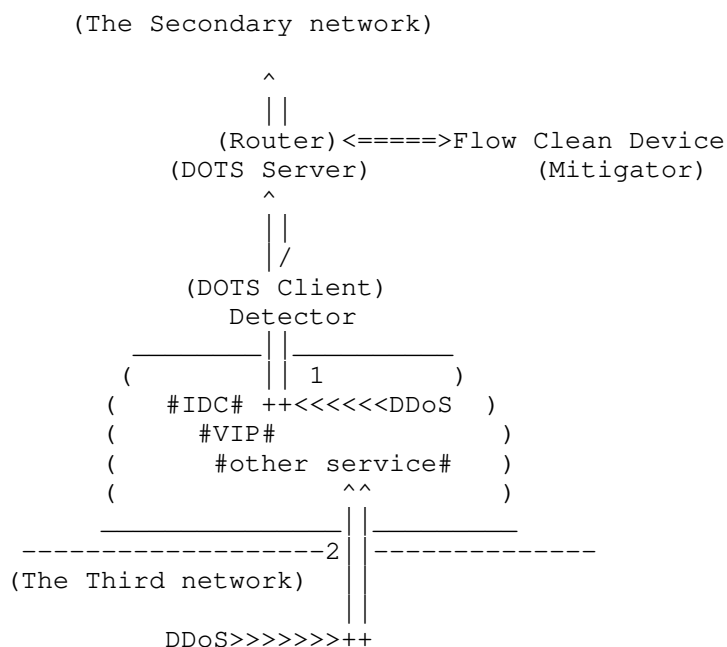


Figure 3: DOTS Agents Deployment

When DDoS Attack path case 1 occurs, the DOTS client in the same network will send mitigation requests to DOTS server which installed in the same area within export router.

When DDoS Attack path case 2 occurs, the Dots server in the Third network export router will receive mitigation request. If the first level of protection is not effective enough, the DOTS server in the upper network will also receive mitigation requests.

If attacks on the same attack target are found both in adjacent areas, there are two strategies for the mitigators' selection, then found the best mitigation node for different scenes.

- o Near Attack Source Mitigation(NASM), NASM means that the mitigation is performed closest to the source of the attack, this usually happens at the entrance to the edge of the network. This approach can block attack flow at the source and protect network bandwidth maximumly, but requires the ability to operate the entire network. This principle is more suitable for large-traffic attack mitigation.
- o Near Attack Target Mitigation(NATM), NATM means that the mitigation is performed closest to the attack target, This is the easiest and most direct way, but it will cause the attack flow long-distance transmission, occupy the bandwidth along the link, more likely to cause link congestion. This principle is more suitable for low-traffic attack mitigation.

Normally, The lower network the target in, the easier it is to alert. Because the higher network the attack target in, the greater the bandwidth of the pipeline. When multiple mitigators need to work together, then need orchestrator to take on the role for scheduling. Because the importance of the orchestrator, it is suggested to consider backup mechanisms or heartbeat technology to ensure continuity and security.

How does DOTS client can find DOTS servers, we can reference the DOTS server discovery draft[I-D.ietf-dots-server-discovery], Static configuration or dynamic discovery depends on the actual scenario and the size of the network.

4.2. DOTS Agents interfaces

In the dots use case draft[I-D.ietf-dots-use-cases], it is says the orchestrator analyses the various information it receives from DDoS telemetry system, and initiates one or multiple DDoS mitigation strategies. In the telemetry draft, all the telemetry informations are contained and some parameters can be used to make decisions. This section made a discussion on which attributes could be used in orchestrator for scheduling.

We suggest orchestrator has three capabilities and reuse the method of registration and notification in signal channel to know all the related mitigators capability and residue capability:

- 1.Can get the netflow/ipfix collector's telemetry informations.

2.Can get the capabilities of each mitigator, it means the initial capacity, this means that with each addition of mitigator there needs to be a protocol that can push this information to orchestrator, we recommend using DOTS signal channel to transfer initial capacity.

3.When mitigation finished, mitigator can inform orchestrator that mitigation is finished and capacity has been released, also we recommend using DOTS signal channel to transfer.

4.2.1. Bandwidth consuming attack

The following parameters will be required by orchestrator:

- o top-talker
- o source-prefix
- o total-traffic
- o total-attack-traffic
- o total-pipe-capability

The recommended approach here is to redirect traffic and flow cleaning.

4.2.2. Host resource consuming attack

The following parameters will be required by orchestrator:

- o top-talker
- o source-prefix

The recommended approach here is to use router for disposition.

5. DOTS server deployment between ISPs

Because of global connectivity, the coexistence of different operators is very common, coordination between operators across networks is very important. Interdomain attacks occur frequently, We recommend deploying the DOTS server at the access point.

- o DDoS attack from one of other ISPs, for example, ISP A received DDoS attack from ISP B or ISP C, then dots server inside ISP B or ISP C will receive the mitigation requests.

- o DDoS attacks from two or more of other ISPs, for example, ISP A and ISP B both start ddos attack to ISP C, then dots server A and dots server B will both receive mitigation request from dots client C.

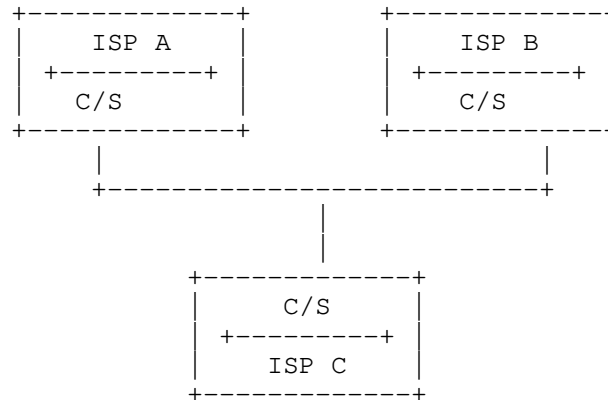
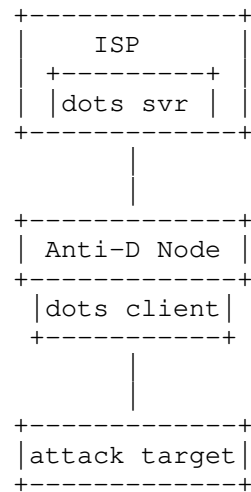


Figure 4: DOTS Agents Deployment between ISPs

When an DDoS attack occurs, depending on the direction of the attack, the corresponding server is required for mitigation, DOTS server can use call home to find the source of the DDoS attacks[I-D.ietf-dots-signal-call-home]

6. DOTS server deployment for Enterprise

In addition to operators taking advantage of the pipeline to make a contribution to DDoS attack mitigation, there are also enterprise-level DDoS attack mitigation solutions. It's usually a cloud service and a large number of distributed nodes are deployed to protect their customers from DDoS attack, customers' websites can be hidden behind the nodes, usually the internet game companies and the live streaming company will choose this way.



*Anti-D is for Anti-DDoS

Figure 5: Deployment for Enterprise and ISP

When enterprise-level anti-DDoS nodes are unable to mitigate the DDoS attack, they can trigger DOTS client which integrated in the Anti-D Node to send mitigation request to ISP's DOTS server.

7. Security Considerations

TBD

8. IANA Considerations

TBD

9. Acknowledgement

TBD

10. References

10.1. Normative References

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

- [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>>.

10.2. Informative References

- [I-D.ietf-dots-architecture]
Mortensen, A., Reddy.K, T., Andreasen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-18 (work in progress), March 2020.
- [I-D.ietf-dots-multihoming]
Boucadair, M., Reddy.K, T., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-04 (work in progress), May 2020.
- [I-D.ietf-dots-requirements]
Mortensen, A., K, R., and R. Moskowitz, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-22 (work in progress), March 2019.
- [I-D.ietf-dots-server-discovery]
Boucadair, M. and T. Reddy.K, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Agent Discovery", draft-ietf-dots-server-discovery-10 (work in progress), February 2020.
- [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-08 (work in progress), March 2020.
- [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.
- [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-23 (work in progress), May 2020.

Authors' Addresses

Meiling Chen
China Mobile

32, Xuanwumen West

BeiJing

,
BeiJing

100053

China

Email:

chenmeiling@chinamobile.com

Li Su
China Mobile

32, Xuanwumen West

BeiJing

100053

China

Email:

suli@chinamobile.com

DOTS
Internet-Draft
Intended status: Informational
Expires: September 3, 2020

Y. Hayashi
NTT
M. Chen
CMCC
March 2, 2020

Use Cases for DDoS Open Threat Signaling (DOTS) Telemetry
draft-hayashi-dots-telemetry-use-cases-00

Abstract

Denial-of-service Open Threat Signaling (DOTS) Telemetry enriches the base DOTS protocols to assist the mitigator in using efficient DDoS-attack-mitigation techniques in a network. This document presents sample use cases for DOTS Telemetry: what components are deployed in the network, how they cooperate, and what information is exchanged to effectively use these techniques.

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 September 3, 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	2
2. Terminology	3
3. Use Cases	3
3.1. DDoS Mitigation Based on Attack Traffic Bandwidth	3
3.1.1. Mitigating Attack Flow of Top-talker Preferentially	3
3.1.2. Optimal DMS Selection for Mitigation	5
3.1.3. Best-path Selection for Redirection	6
3.1.4. Short but Extreme Volumetric Attack Mitigation	8
3.2. DDoS Mitigation Based on Attack Type	9
3.2.1. Selecting Mitigation Technique	9
3.3. Training Flow Collector Using Supervised Machine Learning	11
4. Security Considerations	12
5. IANA Considerations	12
6. Acknowledgement	12
7. References	12
7.1. Normative References	12
7.2. Informative References	13
Authors' Addresses	13

1. Introduction

Denial-of-Service (DDoS), attacks such as volumetric attacks and resource-consumption attacks, are critical threats to be handled by service providers. When such DDoS attacks occur, service providers have to mitigate them immediately to protect or recover their services.

Therefore, for service providers to immediately protect their network services from DDoS attacks, DDoS mitigation needs to be automated. To automate DDoS-attack mitigation, multi-vendor components involved in DDoS-attack detection and mitigation should cooperate and support standard interfaces to communicate.

DDoS Open Threat Signaling (DOTS) is a set of protocols for real-time signaling, threat-handling requests, and data filtering between the multi-vendor elements

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

Furthermore, DOTS Telemetry enriches the DOTS protocols with various telemetry attributes allowing optimal DDoS-attack mitigation

[I-D.ietf-dots-telemetry]. This document presents sample use cases for DOTS Telemetry: what components are deployed in the network, how they cooperate, and what information is exchanged to effectively use attack-mitigation techniques.

2. Terminology

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

In addition, this document uses the following terms:

Top-talker: A top N list of attackers who attack the same target or targets. The list is ordered in terms of a two-tuple bandwidth such as bps or pps.

Supervised Machine Learning: A machine-learning technique that maps an input to an output based on example input-output pairs

3. Use Cases

This section describes DOTS-Telemetry use cases that use attributes included in DOTS Telemetry specifications.

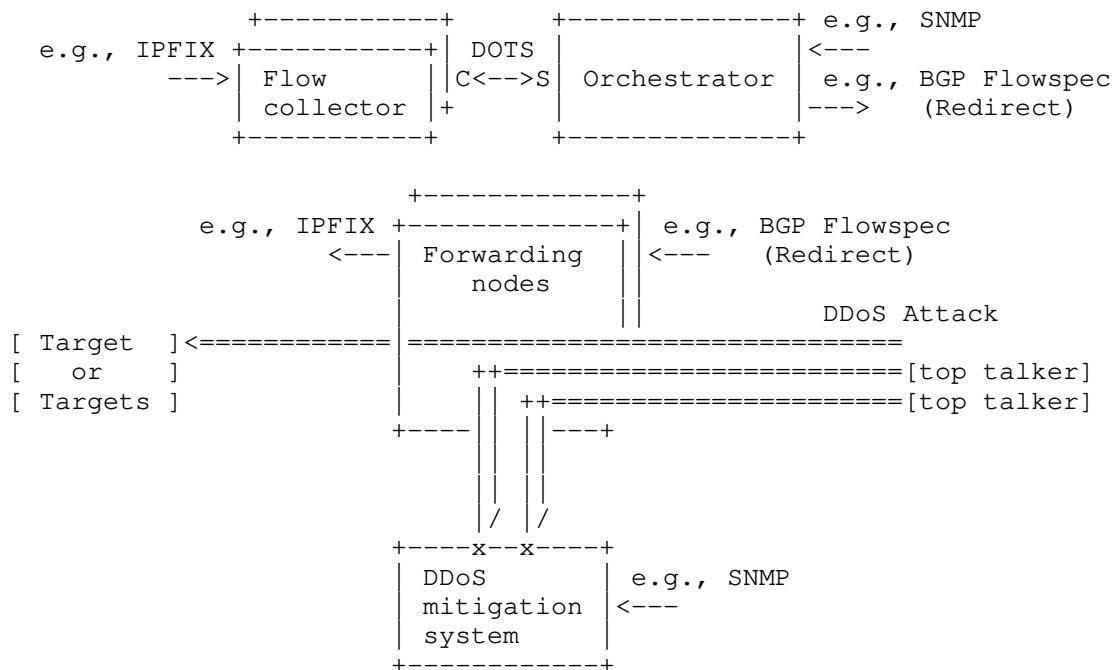
3.1. DDoS Mitigation Based on Attack Traffic Bandwidth

3.1.1. Mitigating Attack Flow of Top-talker Preferentially

Large-scale DDoS attacks, such as amplification attacks, often occur. Some transit providers have to mitigate large-scale DDoS attacks using DMS with limited resources, which is already deployed in their network.

The aim of this use case is to enable transit providers to use their DMS efficiently under volume-based DDoS attacks whose bandwidth is more than the available capacity of the DMS. To enable this, attack traffic of top talkers is redirected to the DMS preferentially by cooperation among forwarding nodes, flow collectors, and orchestrators. Figure 1 gives an overview of this use case.

(Internet Transit Provider)



```
* C is for DOTS client functionality
* S is for DOTS client functionality
```

Figure 1: Mitigating DDoS Attack Flow of Top-talker Preferentially

In this use case, the forwarding nodes always send statistics of traffic flow to the flow collectors by using monitoring functions such as IPFIX[RFC7011]. When DDoS attacks occur, the flow collectors detect attack traffic and send (src_ip, dst_ip, bandwidth)-tuple information of the top talker to the orchestrator using the top-talkers attribute of DOTS Telemetry. The orchestrator then checks the available capacity of DMS by using a network management protocol such as SNMP[RFC3413]. After that, the orchestrator orders forwarding nodes to redirect as much of the top taker's traffic to the DMS as possible by dissemination of flow-specification-rule protocols such as BGP Flowspec[RFC5575].

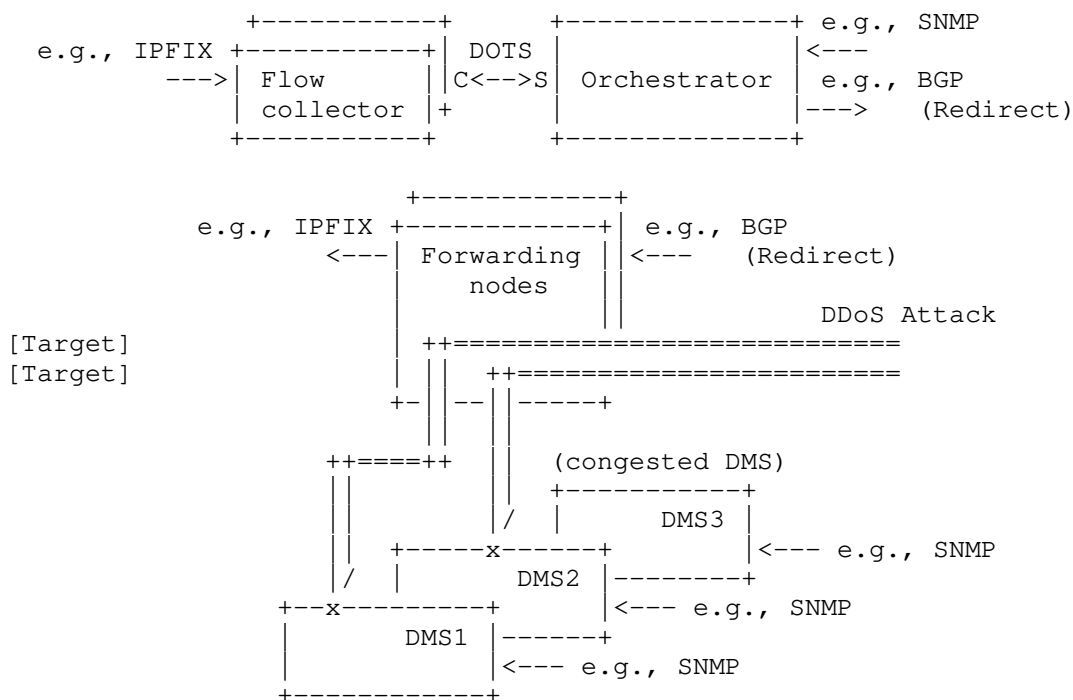
In this case, the flow collector implements a DOTS client while the orchestrator implements a DOTS server.

3.1.2. Optimal DMS Selection for Mitigation

Transit providers, which have a number of DMSs, usually deploy a DMS in various forms to satisfy their requirements; individual or clustered. In both forms, they can identify attributes of the DMSs such as total capacity, available capacity, and the last hop bandwidth.

The aim of this use case is to enable transit providers to select an optimal DMS for mitigation based on the bandwidth of attack traffic, capacity of a DMS, and the last hop bandwidth. Figure 2 gives an overview of this use case.

(Internet Transit Provider)



* C is for DOTS client functionality

* S is for DOTS server functionality

Figure 2: Optimal DMS selection for Mitigation

In this use case, the forwarding nodes always send statistics of traffic flow to the flow collectors by using monitoring functions such as IPFIX[RFC7011]. When DDoS attacks occur, the flow collectors

detect attack traffic and send (dst_ip, bandwidth)-tuple information to the orchestrator using the total-attack-traffic attribute of DOTS Telemetry. The orchestrator then checks the available capacity of the DMSs by using a network management protocol such as SNMP[RFC3413]. After that, the orchestrator chooses optimal DMS which each attack traffic should be redirected. The orchestrator then orders forwarding nodes to redirect the attack traffic to the optimal DMS by a routing protocol such as BGP[RFC4271]. The algorithm of selecting a DMS is out of the scope of this draft.

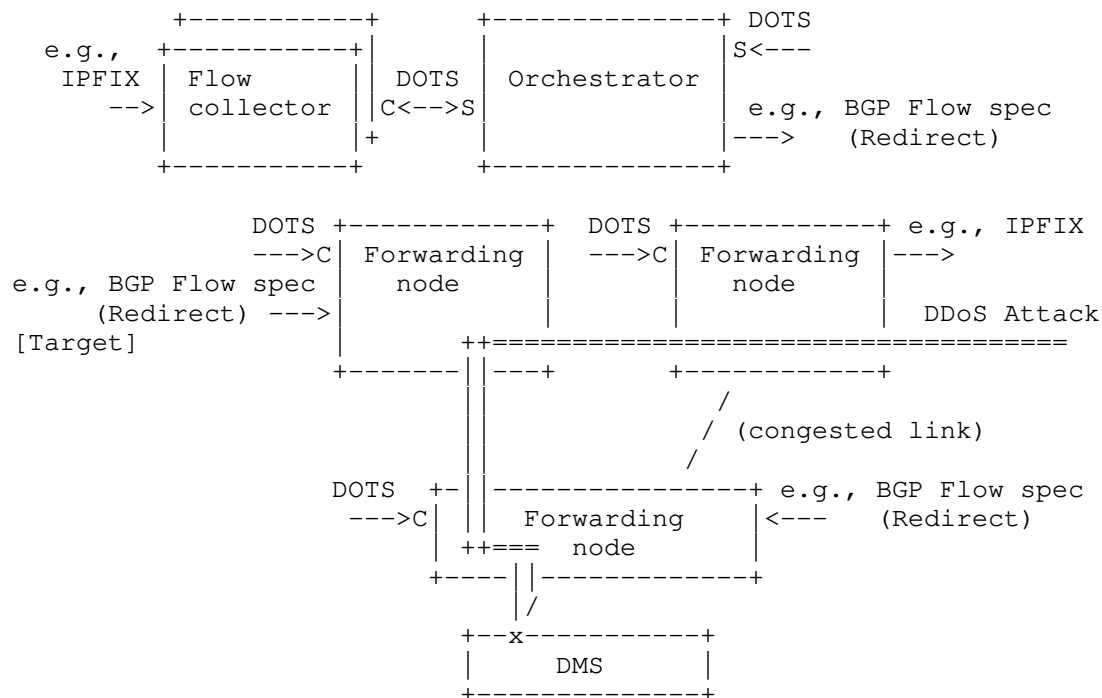
In this case, the flow collector implements a DOTS client while the orchestrator implements a DOTS server.

3.1.3. Best-path Selection for Redirection

A transit-provider network, which adopts a mesh network, has multiple paths to convey attack traffic to a DMS. In this network, attack traffic can be conveyed while avoiding congested links by selecting an available path.

The aim of this use case is to enable transit providers to select an optimal path for redirecting attack traffic to a DMS according to the bandwidth of the attack traffic, total traffic, and total pipe capability. Figure 3 gives an overview of this use case.

(Internet Transit Provider)



* C is for DOTS client functionality
 * S is for DOTS server functionality

Figure 3: Best-path Selection for Redirection

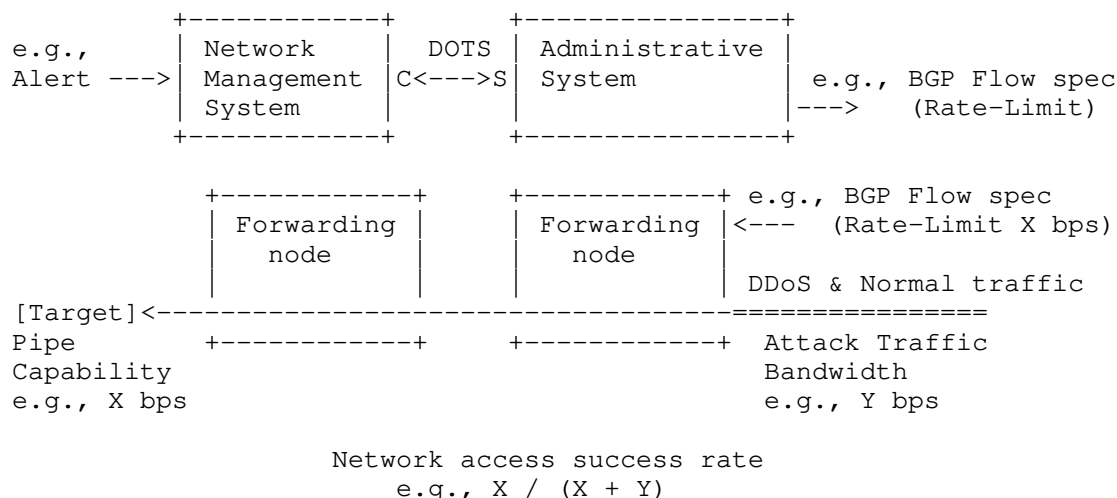
In this use case, the forwarding nodes always send statistics of traffic flow to the flow collectors by using monitoring functions such as IPFIX[RFC7011]. When DDoS attacks occur, the flow collectors detect attack traffic and send (dst_ip, bandwidth)-tuple information to the orchestrator using a total-attack-traffic attribute of DOTS Telemetry. On the other hands, forwarding nodes send bandwidth of total traffic passing the node and total pipe capability to the orchestrator using total-traffic and total-pipe-capability attributes of DOTS Telemetry. The orchestrator then selects an optimal path to which each attack-traffic flow should be redirected. After that, the orchestrator orders forwarding nodes to redirect the attack traffic to the optimal DMS by dissemination of flow-specification-rules protocols such as BGP Flowspec[RFC5575]. The algorithm of selecting a path is out of the scope of this draft.

3.1.4. Short but Extreme Volumetric Attack Mitigation

Short but extreme volumetric attacks, such as pulse wave DDoS attacks, are threats to internet transit provider networks. It is difficult for them to mitigate an attack by DMS by redirecting attack flows because it may cause route flapping in the network. The practical way to mitigate short but extreme volumetric attacks is to offload a mitigation actions to a forwarding node.

The aim of this use case is to enable transit providers to mitigate short but extreme volumetric attacks and estimate the network-access success rate based on the bandwidth of attack traffic and total pipe capability. Figure 4 gives an overview of this use case.

(Internet Transit Provider)



* C is for DOTS client functionality

* S is for DOTS server functionality

Figure 4: Short but Extreme Volumetric Attack Mitigation

In this use case, when DDoS attacks occur, the network management system receives alerts. It then sends the target ip address, pipe capability of the target's link, and bandwidth of the DDoS attack traffic to the administrative system using the target, total-pipe-capability and total-attack-traffic attributes of DOTS Telemetry. After that, the administrative system orders upper forwarding nodes to carry out rate-limit all traffic destined to the target based on the pipe capability by the dissemination of the flow -specification-

rules protocols such as BGP Flowspec[RFC5575]. In addition, the administrative system estimates the network-access success rate of the target, which is calculated by $\text{total pipe capability} / (\text{total pipe capability} + \text{total attack traffic})$.

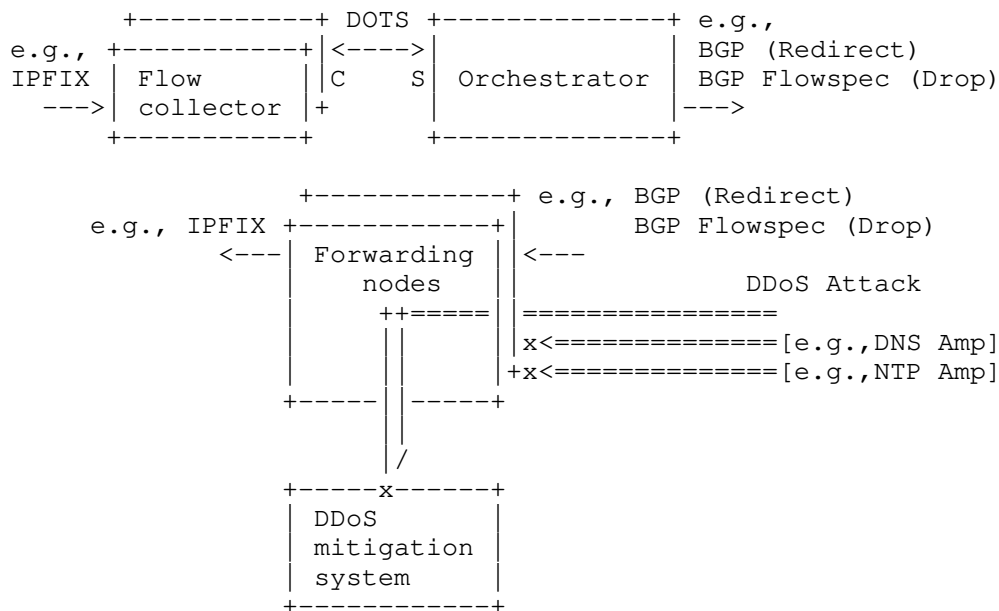
3.2. DDoS Mitigation Based on Attack Type

3.2.1. Selecting Mitigation Technique

Some volumetric attacks, such as amplification attacks, can be detected with high accuracy by checking the layer-3 or layer-4 information of attack packets. These attacks can be detected and mitigated through cooperation among forwarding nodes and flow collectors using IPFIX[RFC7011]. On the other hand, it is necessary to inspect the layer-7 information of attack packets to detect attacks such as DNS Water Torture Attacks. Such attack traffic should be detected and mitigated at a DMS.

The aim of this use case is to enable transit providers to select a mitigation technique based on the type of attack traffic: amplification attack or not. To use such a technique, attack traffic is blocked at forwarding nodes or redirected to a DMS based on attack type through cooperation among forwarding nodes, flow collectors, and an orchestrator. Figure 5 gives an overview of this use case.

(Internet Transit Provider)



* C is for DOTS client functionality

* S is for DOTS server functionality

Figure 5: DDoS Mitigation Based on Attack Type

In this use case, the forwarding nodes send statistics of traffic flow to the flow collectors by using a monitoring function such as IPFIX[RFC7011]. When DDoS attacks occur, the flow collectors detect attack traffic and send (dst_ip, src_port, attack_type)-tuple information to the orchestrator the using attack-name attribute of DOTS Telemetry. The orchestrator then orders forwarding nodes to block the (dst_ip, src_port)-tuple flow of amp attack traffic by dissemination of flow-specification-rule protocols such as BGP Flowspec[RFC5575]. On the other hand, the orchestrator orders forwarding nodes to redirect other traffic than the amp attack traffic by a routing protocol such as BGP[RFC4271].

In this case, the flow collector implements a DOTS client while the orchestrator implements a DOTS server.

3.3. Training Flow Collector Using Supervised Machine Learning

DDoS detection based on monitoring functions, such as IPFIX[RFC7011], is a lighter weight method of detecting DDoS attacks than DMSSs in internet transit provider networks. On the other hand, DDoS detection based on the DMSSs is a more accurate method of detecting DDoS attacks than DDoS detection based on flow monitoring.

The aim of this use case is to increase flow collector's detection accuracy by carrying out supervised machine-learning techniques based on the detection results of the DMSs. To use such a technique, forwarding nodes, flow collector, and a DMS should cooperate. Figure 5 gives an overview of this use case.

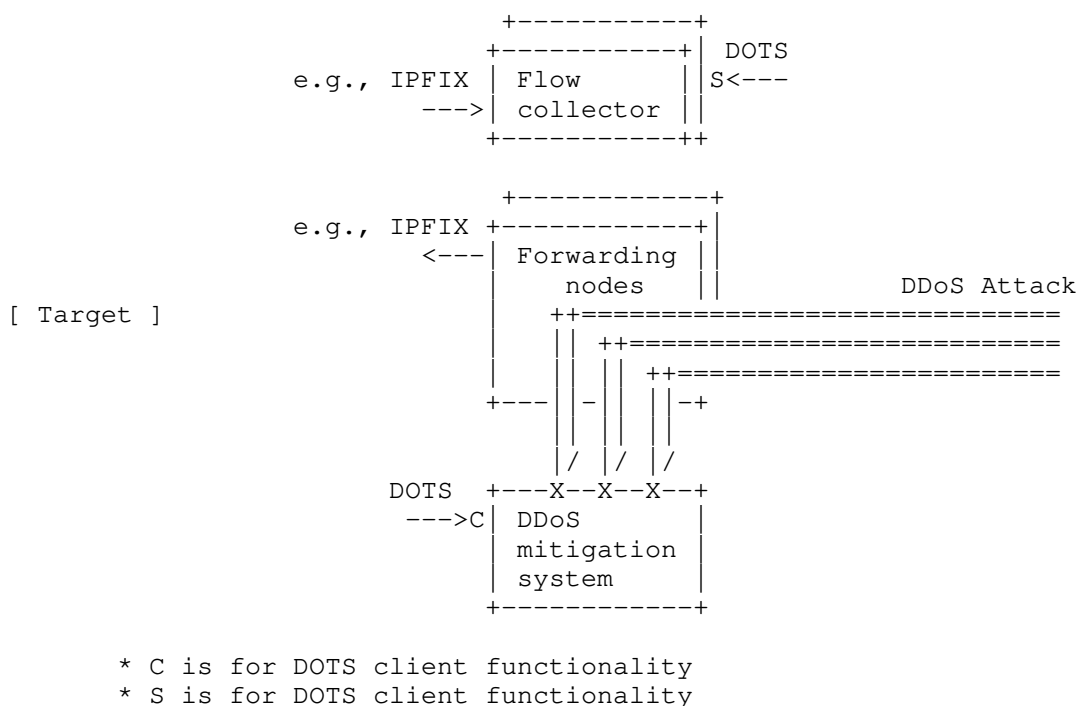


Figure 6: Training Flow Collector Using Supervised Machine Learning

In this use case, the forwarding nodes always send statistics of traffic flow to the flow collectors by using monitoring functions such as IPFIX[RFC7011]. When DDoS attacks occur, DDoS orchestration use case[I-D.ietf-dots-use-cases] is carried out and the DMS mitigates all attack traffic destined for a target. The DDoS-mitigation system reports the (src_ip, dst_ip)-tuple information of

the top talker to the orchestrator the using top-talkers attribute of DOTS Telemetry.

After mitigating a DDoS attack, the flow collector attaches teacher labels to the statistics of traffic flow based on the reports. The label shows normal traffic or attack name. The flow collector then carries out supervised machine learning to increase its detection accuracy, setting the statistics as an explanatory variable and setting the labels as an objective variable.

In this case, the DMS implements a DOTS client while the flow collector implements a DOTS server.

4. Security Considerations

TBD

5. IANA Considerations

This document does not require any action from IANA.

6. Acknowledgement

The authors would like to thank among others brabra...

7. References

7.1. Normative References

[I-D.ietf-dots-telemetry]

Boucadair, M., Reddy, K. T., Doron, E., and C. Chenmeiling, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry", draft-ietf-dots-telemetry-02 (work in progress), February 2020.

[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.

[RFC3413]

Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, DOI 10.17487/RFC3413, December 2002, <<https://www.rfc-editor.org/info/rfc3413>>.

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [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>>.

7.2. Informative References

- [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-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.

Authors' Addresses

Yuhei Hayashi
NTT
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: yuuhei.hayashi@gmail.com

Meiling Chen
CMCC
32, Xuanwumen West
BeiJing, BeiJing 100053
China

Email:
chenmeiling@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 30, 2020

M. Boucadair
Orange
T. Reddy
McAfee
W. Pan
Huawei Technologies
May 29, 2020

Multi-homing Deployment Considerations for Distributed-Denial-of-Service
Open Threat Signaling (DOTS)
draft-ietf-dots-multihoming-04

Abstract

This document discusses multi-homing considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS). The goal is to provide some guidance for DOTS clients/gateways when multihomed.

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 November 30, 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	2
2. Requirements Language	4
3. Terminology	4
4. Multi-Homing Scenarios	4
4.1. Residential Single CPE	5
4.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs	5
4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs	6
4.4. Multi-homed Enterprise with the Same ISP	7
5. DOTS Multi-homing Deployment Considerations	7
5.1. Residential CPE	8
5.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs	9
5.3. Multi-Homed Enterprise: Multiple CPEs, Multiple Upstream ISPs	11
5.4. Multi-Homed Enterprise: Single ISP	12
6. Security Considerations	13
7. IANA Considerations	13
8. Acknowledgements	13
9. References	13
9.1. Normative References	13
9.2. Informative References	14
Authors' Addresses	15

1. Introduction

In many deployments, it may not be possible for a network to determine the cause of a distributed Denial-of-Service (DoS) attack [RFC4732]. Rather, the network may just realize that some resources seem to be under attack. To improve such situation, the IETF is specifying the DDoS Open Threat Signaling (DOTS) [I-D.ietf-dots-architecture] architecture, where a DOTS client can inform a DOTS server that the network is under a potential attack and that appropriate mitigation actions are required. Indeed, because the lack of a common method to coordinate a real-time response among involved actors and network domains jeopardizes the efficiency of DDoS attack mitigation actions, the DOTS protocol is meant to carry requests for DDoS attack mitigation, thereby reducing the impact of an attack and leading to more efficient responsive actions. [I-D.ietf-dots-use-cases] identifies a set of scenarios for DOTS; most of these scenarios involve a Customer Premises Equipment (CPE).

The high-level DOTS architecture is illustrated in Figure 1 ([I-D.ietf-dots-architecture]):

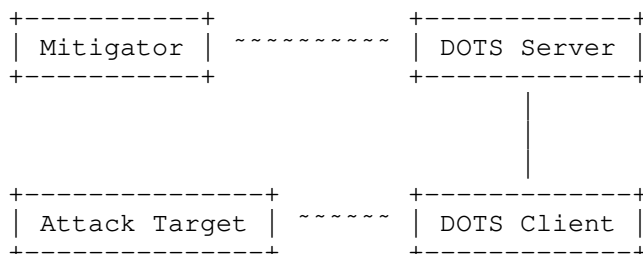


Figure 1: Basic DOTS Architecture

[I-D.ietf-dots-architecture] specifies that the DOTS client may be provided with a list of DOTS servers; each of these servers is associated with one or more IP addresses. These addresses may or may not be of the same address family. The DOTS client establishes one or more DOTS sessions by connecting to the provided DOTS server(s) addresses.

DOTS may be deployed within networks that are connected to one single upstream provider. It can also be enabled within networks that are multi-homed. The reader may refer to [RFC3582] for an overview of multi-homing goals and motivations. This document discusses DOTS multi-homing considerations. Specifically, the document aims to:

1. Complete the base DOTS architecture with multi-homing specifics. Those specifics need to be taken into account because:
 - * Send a DOTS mitigation request to an arbitrary DOTS server won't help mitigating a DDoS attack.
 - * Blindly forking all DOTS mitigation requests among all available DOTS servers is suboptimal.
 - * Sequentially contacting DOTS servers may increase the delay before a mitigation plan is enforced.
2. Identify DOTS deployment schemes in a multi-homing context, where DOTS services can be offered by all or a subset of upstream providers.
3. Sketch guidelines and recommendations for placing DOTS requests in multi-homed networks, e.g.,:
 - * Select the appropriate DOTS server(s).

- * Identify cases where anycast is not recommended.

This document adopts the following methodology:

- o Identify and extract viable deployment candidates from [I-D.ietf-dots-use-cases].
- o Augment the description with multi-homing technicalities, e.g.,
 - * One vs. multiple upstream network providers
 - * One vs. multiple interconnect routers
 - * Provider-Independent (PI) vs. Provider-Aggregatable (PA) IP addresses
- o Describe the recommended behavior of DOTS clients and gateways for each case.

Multi-homed DOTS agents are assumed to make use of the protocols defined in [I-D.ietf-dots-signal-channel] and [I-D.ietf-dots-data-channel]; no specific extension is required to the base DOTS protocols for deploying DOTS in a multi-homed context.

2. Requirements Language

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.

3. Terminology

This document makes use of the terms defined in [I-D.ietf-dots-architecture] and [RFC4116].

IP indifferently refers to IPv4 or IPv6.

4. Multi-Homing Scenarios

This section describes some multi-homing scenarios that are relevant to DOTS. In the following sub-sections, only the connections of border routers are shown; internal network topologies are not elaborated.

This section distinguishes between residential CPEs vs. enterprise CPEs because PI addresses may be used for enterprises while this is not the current practice for residential CPEs.

4.1. Residential Single CPE

The scenario shown in Figure 2 is characterized as follows:

- o The home network is connected to the Internet using one single CPE (Customer Premises Equipment).
- o The CPE is connected to multiple provisioning domains (i.e., both fixed and mobile networks). Provisioning domain (PvD) is explained in [RFC7556].
- o Each of these provisioning domains assigns IP addresses/prefixes to the CPE and provides additional configuration information such as a list of DNS servers, DNS suffixes associated with the network, default gateway address, and DOTS server's name [I-D.ietf-dots-server-discovery]. These addresses/prefixes are assumed to be Provider-Aggregatable (PA).
- o Because of ingress filtering, packets forwarded by the CPE towards a given provisioning domain must be sent with a source IP address that was assigned by that domain [RFC8043].

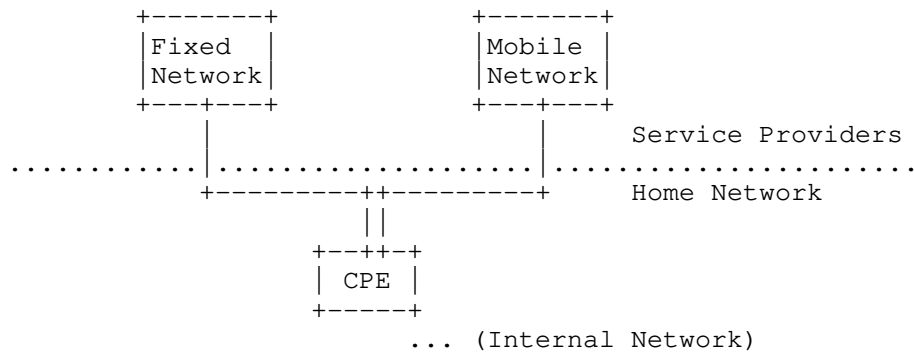


Figure 2: Typical Multi-homed Residential CPE

4.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs

The scenario shown in Figure 3 is characterized as follows:

- o The enterprise network is connected to the Internet using one single router.

- o That router is connected to multiple provisioning domains (i.e., managed by distinct administrative entities).

Unlike the previous scenario, two sub-cases can be considered for an enterprise network with regards to assigned addresses:

1. PI addresses/prefixes: The enterprise is the owner of the IP addresses/prefixes; the same address/prefix is then used when establishing communications over any of the provisioning domains.
2. PA addresses/prefixes: Each of the provisioning domains assigns IP addresses/prefixes to the enterprise network.

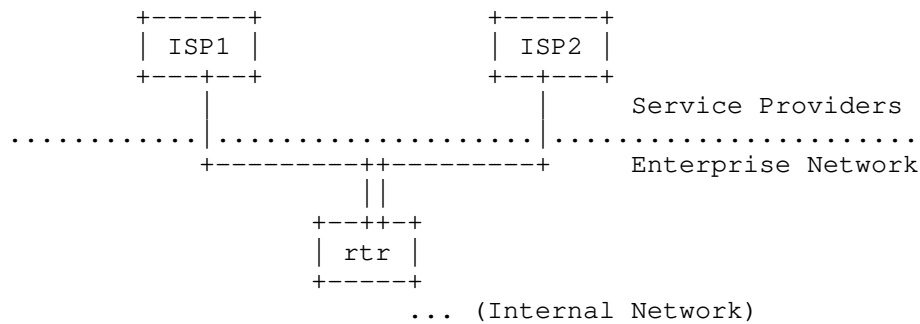


Figure 3: Multi-homed Enterprise Network (Single CPE connected to Multiple Networks)

4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

This scenario is similar to the one described in Section 4.2; the main difference is that dedicated routers are used to connect to each provisioning domain.

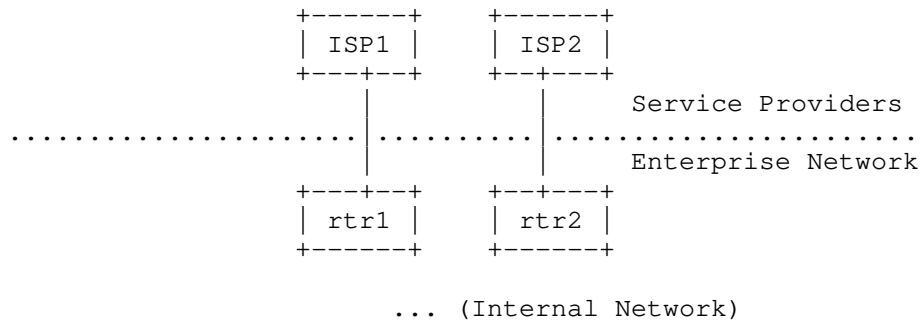


Figure 4: Multi-homed Enterprise Network (Multiple CPEs, Multiple ISPs)

4.4. Multi-homed Enterprise with the Same ISP

This scenario is a variant of Section 4.2 and Section 4.3 in which multi-homing is supported by the same ISP (i.e., same provisioning domain).

Editor's Note: The use of anycast addresses is to be consistently discussed.

5. DOTS Multi-homing Deployment Considerations

Table 1 provides some sample, non-exhaustive, deployment schemes to illustrate how DOTS agents may be deployed for each of the scenarios introduced in Section 4.

Scenario	DOTS client	DOTS gateway
Residential CPE	CPE	N/A
Single CPE, Multiple provisioning domains	internal hosts or CPE	CPE
Multiple CPEs, Multiple provisioning domains	internal hosts or all CPEs (rtr1 and rtr2)	CPEs (rtr1 and rtr2)
Multi-homed enterprise, Single provisioning domain	internal hosts or all CPEs (rtr1 and rtr2)	CPEs (rtr1 and rtr2)

Table 1: Sample Deployment Cases

These deployment schemes are further discussed in the following sub-sections.

5.1. Residential CPE

Figure 5 depicts DOTS sessions that need to be established between a DOTS client (C) and two DOTS servers (S1, S2) within the context of the scenario described in Section 4.1.

For each provisioning domain, the DOTS client MUST resolve the DOTS server's name provided by a provisioning domain ([I-D.ietf-dots-server-discovery]) using the DNS servers learned from the respective provisioning domain. IPv6-capable DOTS clients MUST use the source address selection algorithm defined in [RFC6724] to select the candidate source addresses to contact each of these DOTS servers. DOTS sessions MUST be established and maintained with each of the DOTS servers because the mitigation scope of these servers is restricted. The DOTS client SHOULD use the certificate provisioned by a provisioning domain to authenticate itself to the DOTS server provided by the same provisioning domain.

When conveying a mitigation request to protect the attack target(s), the DOTS client among the DOTS servers available MUST select a DOTS server whose network has assigned the prefixes from which target prefixes and target IP addresses are derived. This implies that if no appropriate DOTS server is found, the DOTS client MUST NOT send the mitigation request to any DOTS server.

For example, a mitigation request to protect target resources bound to a PA IP address/prefix cannot be satisfied by a provisioning domain another domain than the one that owns those addresses/prefixes. Consequently, if a CPE detects a DDoS attack that spreads over all its network attachments, it MUST contact both DOTS servers for mitigation purposes. Nevertheless, if the DDoS attack is received from one single network, then only the DOTS server of that network MUST be contacted.

The DOTS client MUST be able to associate a DOTS server with each provisioning domain. For example, if the DOTS client is provisioned with S1 using DHCP when attaching to a first network and with S2 using Protocol Configuration Option (PCO) when attaching to a second network, the DOTS client must record the interface from which a DOTS server was provisioned. DOTS signaling session to a given DOTS server must be established using the interface from which the DOTS server was provisioned.

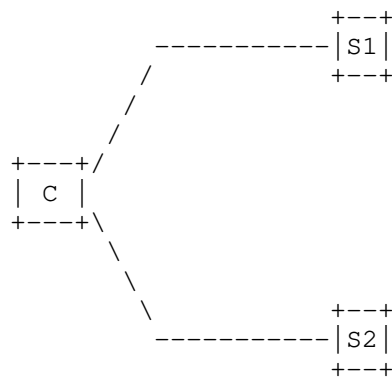


Figure 5: DOTS Associations for a Multihomed Residential CPE

5.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs

Figure 6 illustrates a first set of DOTS associations that can be established with a DOTS gateway, which is enabled within the context of the scenario described in Section 4.2. This deployment is characterized as follows:

- o One of more DOTS clients are enabled in hosts located in the internal network.
- o A DOTS gateway is enabled to aggregate and then relay the requests towards upstream DOTS servers.

When PA addresses/prefixes are in use, the same considerations discussed in Section 5.1 need to be followed by the DOTS gateway to contact its DOTS server(s). The DOTS gateways can be reachable from DOTS clients by using an unicast address or an anycast address.

Nevertheless, when PI addresses/prefixes are assigned, the DOTS gateway MUST send mitigation requests to all its DOTS servers. Otherwise, the attack traffic may still be delivered via the ISP which hasn't received the mitigation request.

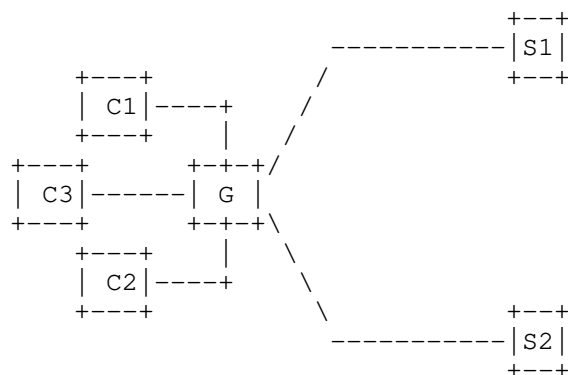


Figure 6: Multiple DOTS Clients, Single DOTS Gateway, Multiple DOTS Servers

An alternate deployment model is depicted in Figure 7. This deployment assumes that:

- o One or more DOTS clients are enabled in hosts located in the internal network. These DOTS clients may use [I-D.ietf-dots-server-discovery] to discover their DOTS server(s).
- o These DOTS clients communicate directly with upstream DOTS servers.

If PI addresses/prefixes are in use, the DOTS client MUST send a mitigation request to all the DOTS servers. The use of anycast addresses to reach the DOTS servers is NOT RECOMMENDED.

If PA addresses/prefixes are used, the same considerations discussed in Section 5.1 need to be followed by the DOTS clients. Because DOTS clients are not embedded in the CPE and multiple addresses/prefixes may not be assigned to the DOTS client (typically in an IPv4 context), some issues arise to steer traffic towards the appropriate DOTS server by using the appropriate source IP address. These complications discussed in [RFC4116] are not specific to DOTS.

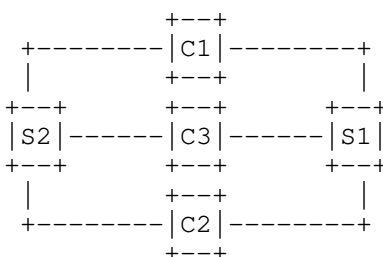


Figure 7: Multiple DOTS Clients, Multiple DOTS Servers

Another deployment approach is to enable many DOTS clients; each of them is responsible for handling communications with a specific DOTS server (see Figure 8).

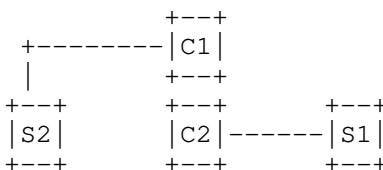


Figure 8: Single Homed DOTS Clients

Each DOTS client SHOULD be provided with policies (e.g., a prefix filter that will be against DDoS detection alarms) that will trigger DOTS communications with the DOTS servers. Such policies will help the DOTS client to select the appropriate destination DOTS server.

The CPE MUST select the appropriate source IP address when forwarding DOTS messages received from an internal DOTS client. If anycast addresses are used to reach DOTS servers, the CPE may not be able to select the appropriate provisioning domain to which the mitigation request should be forwarded. As a consequence, the request may not be forwarded to the appropriate DOTS server.

5.3. Multi-Homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

The deployments depicted in Figures 7 and 8 also apply to the scenario described in Section 4.3. One specific problem for this scenario is to select the appropriate exit router when contacting a given DOTS server.

An alternative deployment scheme is shown in Figure 9:

- o DOTS clients are enabled in hosts located in the internal network.

- o A DOTS gateway is enabled in each CPE (rtr1, rtr2).
- o Each of these DOTS gateways communicates with the DOTS server of the provisioning domain.

When PI addresses/prefixes are used, DOTS clients MUST contact all the DOTS gateways to send a DOTS message. DOTS gateways will then relay the request to the DOTS server. Note that the use of anycast addresses is NOT RECOMMENDED to establish DOTS sessions between DOTS clients and DOTS gateways.

When PA addresses/prefixes are used, but no filter rules are provided to DOTS clients, the latter MUST contact all DOTS gateways simultaneously to send a DOTS message. Upon receipt of a request by a DOTS gateway, it MUST check whether the request is to be forwarded upstream (if the target IP prefix is managed by the upstream server) or rejected.

When PA addresses/prefixes are used, but specific filter rules are provided to DOTS clients using some means that are out of scope of this document, the clients MUST select the appropriate DOTS gateway to reach. The use of anycast addresses is NOT RECOMMENDED to reach DOTS gateways.

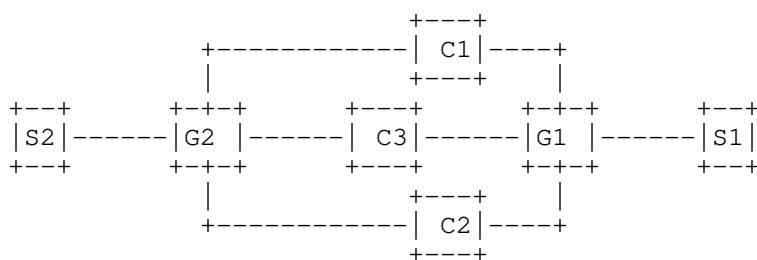


Figure 9: Multiple DOTS Clients, Multiple DOTS Gateways, Multiple DOTS Servers

5.4. Multi-Homed Enterprise: Single ISP

The key difference of the scenario described in Section 4.4 compared to the other scenarios is that multi-homing is provided by the same ISP. Concretely, that ISP can decide to provision the enterprise network with:

1. The same DOTS server for all network attachments.

2. Distinct DOTS servers for each network attachment. These DOTS servers need to coordinate when a mitigation action is received from the enterprise network.

In both cases, DOTS agents enabled within the enterprise network MAY decide to select one or all network attachments to send DOTS mitigation requests.

6. Security Considerations

DOTS-related security considerations are discussed in Section 4 of [I-D.ietf-dots-architecture].

DOTS clients should control the information that they share with peer DOTS servers. For example, if a DOTS client maintains DOTS associations with specific DOTS servers per interconnection link, the DOTS client should not leak information specific to a given link to DOTS servers not authorized to mitigate attacks received on that link. Whether this constraint is relaxed is deployment specific and must be subject to explicit consent from the DOTS client domain administrator.

7. IANA Considerations

This document does not require any action from IANA.

8. Acknowledgements

Thanks to Roland Dobbins, Nik Teague, Jon Shallow, Dan Wing, and Christian Jacquenet for sharing their comments on the mailing list.

Thanks to Kirill Kasavchenko for the comments.

9. References

9.1. Normative References

[I-D.ietf-dots-architecture]

Mortensen, A., Reddy, K., T., Andreassen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-18 (work in progress), March 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>>.

- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [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>>.

9.2. Informative References

- [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-server-discovery]
Boucadair, M. and T. Reddy.K, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Agent Discovery", draft-ietf-dots-server-discovery-10 (work in progress), February 2020.
- [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.
- [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-21 (work in progress), May 2020.
- [RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, DOI 10.17487/RFC3582, August 2003, <<https://www.rfc-editor.org/info/rfc3582>>.
- [RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, DOI 10.17487/RFC4116, July 2005, <<https://www.rfc-editor.org/info/rfc4116>>.

- [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <<https://www.rfc-editor.org/info/rfc4732>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8043] Sarikaya, B. and M. Boucadair, "Source-Address-Dependent Routing and Source Address Selection for IPv6 Hosts: Overview of the Problem Space", RFC 8043, DOI 10.17487/RFC8043, January 2017, <<https://www.rfc-editor.org/info/rfc8043>>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

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

Email: TirumaleswarReddy_Konda@McAfee.com

Wei Pan
Huawei Technologies

Email: william.panwei@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 28 October 2022

M. Boucadair
Orange
T. Reddy.K
Akamai
W. Pan
Huawei Technologies
26 April 2022

Multi-homing Deployment Considerations for Distributed-Denial-of-Service
Open Threat Signaling (DOTS)
draft-ietf-dots-multihoming-13

Abstract

This document discusses multi-homing considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS). The goal is to provide some guidance for DOTS clients and client-domain DOTS gateways when multihomed.

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 28 October 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Terminology	4
4. Multi-Homing Scenarios	5
4.1. Multi-Homed Residential Single CPE	5
4.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs	6
4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs	7
4.4. Multi-homed Enterprise with the Same ISP	7
5. DOTS Multi-homing Deployment Considerations	8
5.1. Residential CPE	8
5.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs	10
5.3. Multi-Homed Enterprise: Multiple CPEs, Multiple Upstream ISPs	12
5.4. Multi-Homed Enterprise: Single ISP	13
6. Security Considerations	14
7. IANA Considerations	14
8. Acknowledgements	14
9. References	15
9.1. Normative References	15
9.2. Informative References	15
Authors' Addresses	16

1. Introduction

In many deployments, it may not be possible for a network to determine the cause of a distributed Denial-of-Service (DoS) attack [RFC4732]. Rather, the network may just realize that some resources appear to be under attack. To help with such situations, the IETF has specified the DDoS Open Threat Signaling (DOTS) architecture [RFC8811], where a DOTS client can inform an upstream DOTS server that its network is under a potential attack and that appropriate mitigation actions are required. The DOTS protocols can be used to coordinate real-time mitigation efforts which can evolve as the attacks mutate, thereby reducing the impact of an attack and leading

to more efficient responsive actions. [RFC8903] identifies a set of scenarios for DOTS; most of these scenarios involve a Customer Premises Equipment (CPE).

The high-level base DOTS architecture is illustrated in Figure 1 ([RFC8811]):

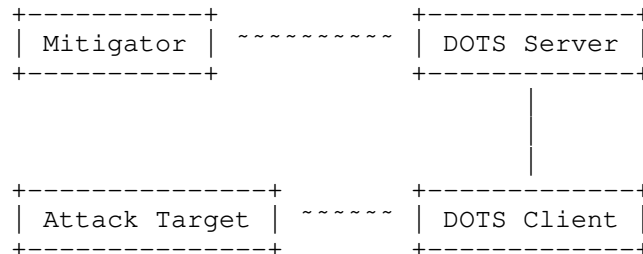


Figure 1: Basic DOTS Architecture

[RFC8811] specifies that the DOTS client may be provided with a list of DOTS servers; each of these servers is associated with one or more IP addresses. These addresses may or may not be of the same address family. The DOTS client establishes one or more DOTS sessions by connecting to the provided DOTS server(s) addresses (e.g., by using [RFC8973]).

DOTS may be deployed within networks that are connected to one single upstream provider. DOTS can also be enabled within networks that are multi-homed. The reader may refer to [RFC3582] for an overview of multi-homing goals and motivations. This document discusses DOTS multi-homing considerations. Specifically, the document aims to:

1. Complete the base DOTS architecture with multi-homing specifics. Those specifics need to be taken into account because:
 - * Sending a DOTS mitigation request to an arbitrary DOTS server will not necessarily help in mitigating a DDoS attack.
 - * Randomly replicating all DOTS mitigation requests among all available DOTS servers is suboptimal.
 - * Sequentially contacting DOTS servers may increase the delay before a mitigation plan is enforced.
2. Identify DOTS deployment schemes in a multi-homing context, where DOTS services can be offered by all or a subset of upstream providers.

3. Provide guidelines and recommendations for placing DOTS requests in multi-homed networks, e.g.,:

- * Select the appropriate DOTS server(s).
- * Identify cases where anycast is not recommended for DOTS.

This document adopts the following methodology:

- * Identify and extract viable deployment candidates from [RFC8903].
- * Augment the description with multi-homing technicalities, e.g.,
 - One vs. multiple upstream network providers
 - One vs. multiple interconnect routers
 - Provider-Independent (PI) vs. Provider-Aggregatable (PA) IP addresses
- * Describe the recommended behavior of DOTS clients and client-domain DOTS gateways for each case.

Multi-homed DOTS agents are assumed to make use of the protocols defined in [RFC9132] and [RFC8783]. This document does not require any specific extension to the base DOTS protocols for deploying DOTS in a multi-homed context.

2. Requirements Language

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.

3. Terminology

This document makes use of the terms defined in [RFC8811], [RFC8612], and [RFC4116]. In particular:

Provider-Aggregatable (PA) addresses: globally-unique addresses assigned by a transit provider to a customer. The addresses are considered "aggregatable" because the set of routes corresponding to the PA addresses are usually covered by an aggregate route set corresponding to the address space operated by the transit provider, from which the assignment was made (Section 2 of [RFC4116]).

Provider-Independent (PI) addresses: globally-unique addresses that are not assigned by a transit provider, but are provided by some other organisation, usually a Regional Internet Registry (RIR) (Section 2 of [RFC4116]).

IP indifferently refers to IPv4 or IPv6.

4. Multi-Homing Scenarios

This section describes some multi-homing scenarios that are relevant to DOTS. In the following subsections, only the connections of border routers are shown; internal network topologies are not elaborated.

A multihomed network may enable DOTS for all or a subset of its upstream interconnection links. In such a case, DOTS servers can be explicitly configured or dynamically discovered by a DOTS client using means such as those discussed in [RFC8973]. These DOTS servers can be owned by the upstream provider, managed by a third-party (e.g., mitigation service provider), or a combination thereof.

If a DOTS server is explicitly configured, it is assumed that an interface is also provided to bind the DOTS service to an interconnection link. If no interface is provided, this means that the DOTS server can be reached via any active interface.

This section distinguishes between residential CPEs vs. enterprise CPEs because PI addresses may be used for enterprises while this is not the current practice for residential CPEs.

In the following subsections, all or a subset of interconnection links are associated with DOTS servers.

4.1. Multi-Homed Residential Single CPE

The scenario shown in Figure 2 is characterized as follows:

- * The home network is connected to the Internet using one single CPE.
- * The CPE is connected to multiple provisioning domains (i.e., both fixed and mobile networks). Provisioning domain (PvD) is explained in [RFC7556].

In a typical deployment scenario, these provisioning domains are owned by the same provider (see Section 1 of [RFC8803]). Such a deployment is meant to seamlessly use both fixed and cellular networks for bonding, faster hand-overs, or better resiliency purposes.

- * Each of these provisioning domains assigns IP addresses/prefixes to the CPE and provides additional configuration information such as a list of DNS servers, DNS suffixes associated with the network, default gateway address, and DOTS server's name [RFC8973]. These addresses/prefixes are assumed to be Provider-Aggregatable (PA).
- * Because of ingress filtering, packets forwarded by the CPE towards a given provisioning domain must be sent with a source IP address that was assigned by that domain [RFC8043].

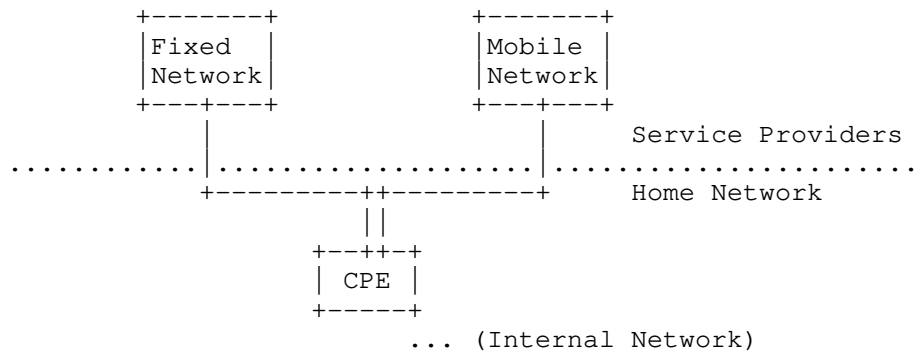


Figure 2: Typical Multi-homed Residential CPE

4.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs

The scenario shown in Figure 3 is characterized as follows:

- * The enterprise network is connected to the Internet using a single router.
- * That router is connected to multiple provisioning domains managed by distinct administrative entities.

Unlike the previous scenario, two sub-cases can be considered for an enterprise network with regards to assigned addresses:

1. PI addresses/prefixes: The enterprise is the owner of the IP addresses/prefixes; the same address/prefix is then used when establishing communications over any of the provisioning domains.

2. PA addresses/prefixes: Each of the provisioning domains assigns IP addresses/prefixes to the enterprise network. These addresses/prefixes are used when communicating over the provisioning domain that assigned them.

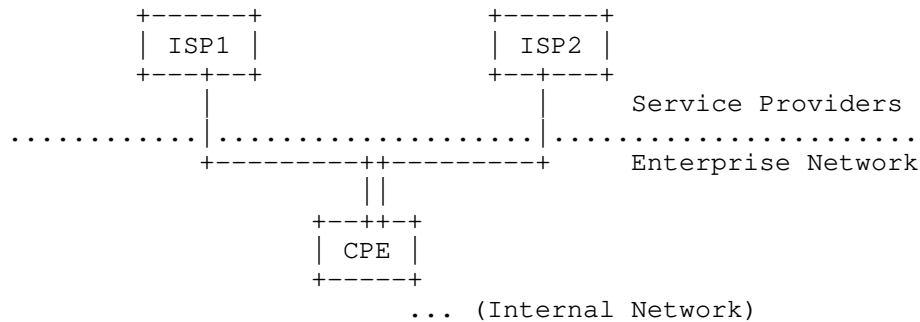


Figure 3: Multi-homed Enterprise Network (Single CPE connected to Multiple Networks)

4.3. Multi-homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

This scenario is similar to the one described in Section 4.2; the main difference is that dedicated routers (CPE1 and CPE2) are used to connect to each provisioning domain.

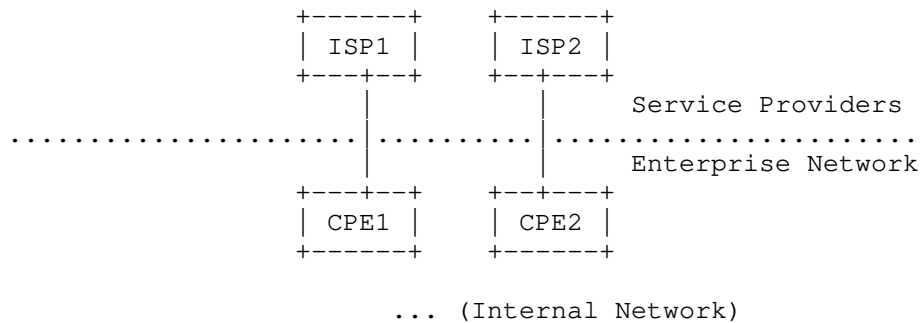


Figure 4: Multi-homed Enterprise Network (Multiple CPEs, Multiple ISPs)

4.4. Multi-homed Enterprise with the Same ISP

This scenario is a variant of Sections 4.2 and 4.3 in which multi-homing is supported by the same ISP (i.e., same provisioning domain).

5. DOTS Multi-homing Deployment Considerations

Table 1 provides some sample, non-exhaustive, deployment schemes to illustrate how DOTS agents may be deployed for each of the scenarios introduced in Section 4.

Scenario	DOTS client	Client-domain DOTS gateway
Residential CPE	CPE	N/A
Single CPE, Multiple provisioning domains	Internal hosts or CPE	CPE
Multiple CPEs, Multiple provisioning domains	Internal hosts or all CPEs (CPE1 and CPE2)	CPEs (CPE1 and CPE2)
Multi-homed enterprise, Single provisioning domain	Internal hosts or all CPEs (CPE1 and CPE2)	CPEs (CPE1 and CPE2)

Table 1: Sample Deployment Cases

These deployment schemes are further discussed in the following subsections.

5.1. Residential CPE

Figure 5 depicts DOTS sessions that need to be established between a DOTS client (C) and two DOTS servers (S1, S2) within the context of the scenario described in Section 4.1. As listed in Table 1, the DOTS client is hosted by the residential CPE.

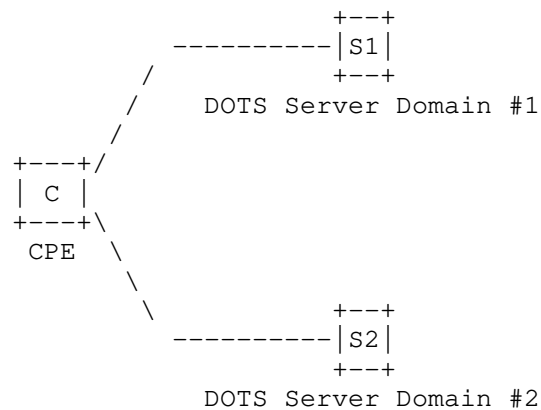


Figure 5: DOTS Associations for a Multihomed Residential CPE

The DOTS client MUST resolve the DOTS server's name provided by each provisioning domain using either the DNS servers learned from the respective provisioning domain or from the DNS servers associated with the interface(s) for which a DOTS server was explicitly configured (Section 4). IPv6-capable DOTS clients MUST use the source address selection algorithm defined in [RFC6724] to select the candidate source addresses to contact each of these DOTS servers. DOTS sessions MUST be established and MUST be maintained with each of the DOTS servers because the mitigation scope of each of these servers is restricted. The DOTS client MUST use the security credentials (a certificate, typically) provided by a provisioning domain to authenticate itself to the DOTS server(s) provided by the same provisioning domain. How such security credentials are provided to the DOTS client is out of the scope of this document. The reader may refer to Section 7.1 of [RFC9132] for more details about DOTS authentication methods.

When conveying a mitigation request to protect the attack target(s), the DOTS client MUST select an available DOTS server whose network has assigned the IP prefixes from which target prefixes/addresses are derived. This implies that if no appropriate DOTS server is found, the DOTS client MUST NOT send the mitigation request to any other available DOTS server.

For example, a mitigation request to protect target resources bound to a PA IP address/prefix cannot be satisfied by a provisioning domain other than the one that owns those addresses/prefixes. Consequently, if a CPE detects a DDoS attack that spreads over all its network attachments, it MUST contact all DOTS servers for mitigation purposes.

The DOTS client MUST be able to associate a DOTS server with each provisioning domain it serves. For example, if the DOTS client is provisioned with S1 using DHCP when attaching to a first network and with S2 using Protocol Configuration Option (PCO) [TS.24008] when attaching to a second network, the DOTS client must record the interface from which a DOTS server was provisioned. A DOTS signaling session to a given DOTS server must be established using the interface from which the DOTS server was provisioned. If a DOTS server is explicitly configured, DOTS signaling with that server must be established via the interfaces that are indicated in the explicit configuration or via any active interface if no interface is configured.

5.2. Multi-Homed Enterprise: Single CPE, Multiple Upstream ISPs

Figure 6 illustrates the DOTS sessions that can be established with a client-domain DOTS gateway (hosted within the CPE as per Table 1), which is enabled within the context of the scenario described in Section 4.2. This deployment is characterized as follows:

- * One or more DOTS clients are enabled in hosts located in the internal network.
- * A client-domain DOTS gateway is enabled to aggregate and then relay the requests towards upstream DOTS servers.

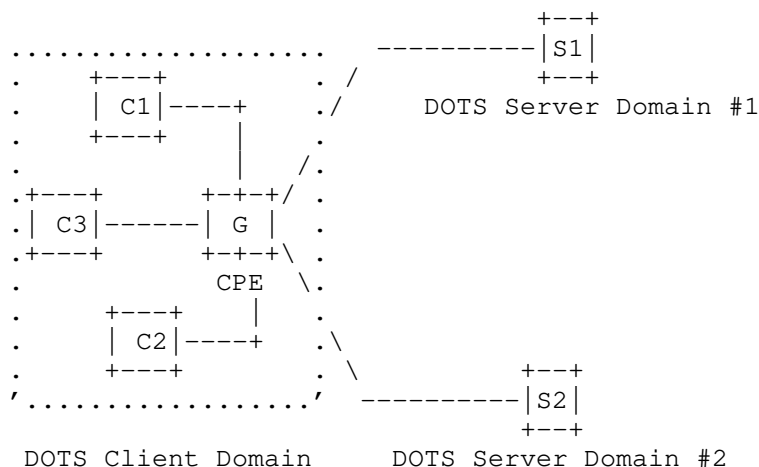


Figure 6: Multiple DOTS Clients, Single DOTS Gateway, Multiple DOTS Servers

When PA addresses/prefixes are in use, the same considerations discussed in Section 5.1 need to be followed by the client-domain DOTS gateway to contact its DOTS server(s). The client-domain DOTS gateways can be reachable from DOTS clients by using a unicast address or an anycast address (Section 3.2.4 of [RFC8811]).

Nevertheless, when PI addresses/prefixes are assigned and absent any policy, the client-domain DOTS gateway SHOULD send mitigation requests to all its DOTS servers. Otherwise, the attack traffic may still be delivered via the ISP that hasn't received the mitigation request.

An alternate deployment model is depicted in Figure 7. This deployment assumes that:

- * One or more DOTS clients are enabled in hosts located in the internal network. These DOTS clients may use [RFC8973] to discover their DOTS server(s).
- * These DOTS clients communicate directly with upstream DOTS servers.

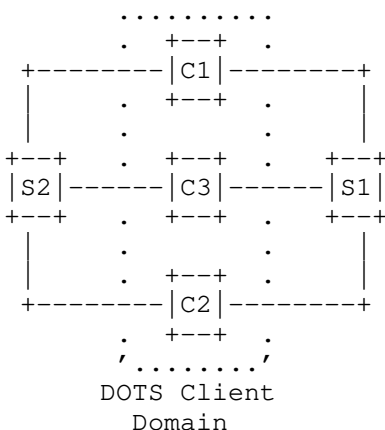


Figure 7: Multiple DOTS Clients, Multiple DOTS Servers

If PI addresses/prefixes are in use, the DOTS client MUST send a mitigation request to all the DOTS servers. The use of the same anycast addresses to reach these DOTS servers is NOT RECOMMENDED. If a well-known anycast address is used to reach multiple DOTS servers, the CPE may not be able to select the appropriate provisioning domain to which the mitigation request should be forwarded. As a consequence, the request may not be forwarded to the appropriate DOTS server.

If PA addresses/prefixes are used, the same considerations discussed in Section 5.1 need to be followed by the DOTS clients. Because DOTS clients are not embedded in the CPE and multiple addresses/prefixes may not be assigned to the DOTS client (typically in an IPv4 context), some issues may arise in how to steer traffic towards the appropriate DOTS server by using the appropriate source IP address. These complications discussed in [RFC4116] are not specific to DOTS.

Another deployment approach is to enable many DOTS clients; each of them is responsible for handling communications with a specific DOTS server (see Figure 8).

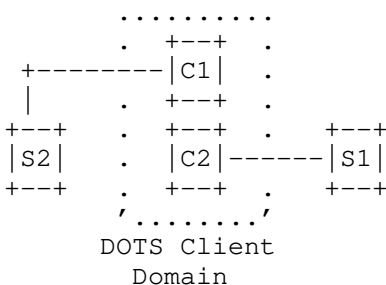


Figure 8: Single Homed DOTS Clients

For both deployments depicted in Figures 7 and 8, each DOTS client SHOULD be provided with policies (e.g., a prefix filter that is used to filter DDoS detection alarms) that will trigger DOTS communications with the DOTS servers. Such policies will help the DOTS client to select the appropriate destination DOTS server. The CPE MUST select the appropriate source IP address when forwarding DOTS messages received from an internal DOTS client.

5.3. Multi-Homed Enterprise: Multiple CPEs, Multiple Upstream ISPs

The deployments depicted in Figures 7 and 8 also apply to the scenario described in Section 4.3. One specific problem for this scenario is to select the appropriate exit router when contacting a given DOTS server.

An alternative deployment scheme is shown in Figure 9:

- * DOTS clients are enabled in hosts located in the internal network.
- * A client-domain DOTS gateway is enabled in each CPE (CPE1 and CPE2 per Table 1).

- * Each of these client-domain DOTS gateways communicates with the DOTS server of the provisioning domain.

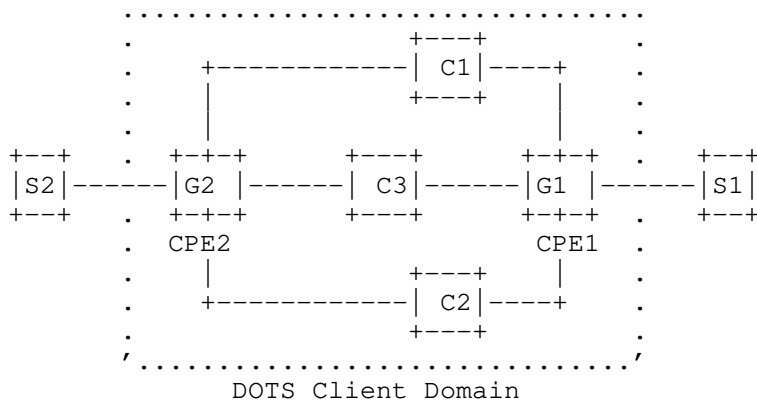


Figure 9: Multiple DOTS Clients, Multiple DOTS Gateways, Multiple DOTS Servers

When PI addresses/prefixes are used, DOTS clients MUST contact all the client-domain DOTS gateways to send a DOTS message. Client-domain DOTS gateways will then relay the request to the DOTS servers as a function of local policy. Note that (same) anycast addresses cannot be used to establish DOTS sessions between DOTS clients and client-domain DOTS gateways because only one DOTS gateway will receive the mitigation request.

When PA addresses/prefixes are used, but no filter rules are provided to DOTS clients, the latter MUST contact all client-domain DOTS gateways simultaneously to send a DOTS message. Upon receipt of a request by a client-domain DOTS gateway, it MUST check whether the request is to be forwarded upstream (if the target IP prefix is managed by the upstream server) or rejected.

When PA addresses/prefixes are used, but specific filter rules are provided to DOTS clients using some means that are out of scope of this document, the clients MUST select the appropriate client-domain DOTS gateway to reach. The use of the same anycast addresses is NOT RECOMMENDED to reach client-domain DOTS gateways.

5.4. Multi-Homed Enterprise: Single ISP

The key difference of the scenario described in Section 4.4 compared to the other scenarios is that multi-homing is provided by the same ISP. Concretely, that ISP can decide to provision the enterprise network with:

- * The same DOTS server for all network attachments.
- * Distinct DOTS servers for each network attachment. These DOTS servers need to coordinate when a mitigation action is received from the enterprise network.

In both cases, DOTS agents enabled within the enterprise network MAY decide to select one or all network attachments to send DOTS mitigation requests.

6. Security Considerations

A set of security threats related to multihoming are discussed in [RFC4218].

DOTS-related security considerations are discussed in Section 4 of [RFC8811].

DOTS clients should control the information that they share with peer DOTS servers. In particular, if a DOTS client maintains DOTS sessions with specific DOTS servers per interconnection link, the DOTS client SHOULD NOT leak information specific to a given link to DOTS servers on different interconnection links that are not authorized to mitigate attacks for that given link. Whether this constraint is relaxed is deployment-specific and must be subject to explicit consent from the DOTS client domain administrator. How to seek for such consent is implementation- and deployment-specific.

7. IANA Considerations

This document does not require any action from IANA.

8. Acknowledgements

Thanks to Roland Dobbins, Nik Teague, Jon Shallow, Dan Wing, and Christian Jacquenet for sharing their comments on the mailing list.

Thanks to Kirill Kasavchenko for the comments.

Thanks to Kathleen Moriarty for the secdir review, Joel Jaeggli for the opsdireview, Mirja Kuhlewind for the tsvar review, and Dave Thaler for the Intdir review.

Many thanks to Roman Danyliw for the careful AD review.

Thanks to Lars Eggert, Robert Wilton, Paul Wouters, Erik Kline, and Eric Vyncke for the IESG review.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [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>>.
- [RFC8811] Mortensen, A., Ed., Reddy, K., T., Ed., Andreasen, F., Teague, N., and R. Compton, "DDoS Open Threat Signaling (DOTS) Architecture", RFC 8811, DOI 10.17487/RFC8811, August 2020, <<https://www.rfc-editor.org/info/rfc8811>>.

9.2. Informative References

- [RFC3582] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, DOI 10.17487/RFC3582, August 2003, <<https://www.rfc-editor.org/info/rfc3582>>.
- [RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, DOI 10.17487/RFC4116, July 2005, <<https://www.rfc-editor.org/info/rfc4116>>.
- [RFC4218] Nordmark, E. and T. Li, "Threats Relating to IPv6 Multihoming Solutions", RFC 4218, DOI 10.17487/RFC4218, October 2005, <<https://www.rfc-editor.org/info/rfc4218>>.
- [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <<https://www.rfc-editor.org/info/rfc4732>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.

- [RFC8043] Sarikaya, B. and M. Boucadair, "Source-Address-Dependent Routing and Source Address Selection for IPv6 Hosts: Overview of the Problem Space", RFC 8043, DOI 10.17487/RFC8043, January 2017, <<https://www.rfc-editor.org/info/rfc8043>>.
- [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>>.
- [RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <<https://www.rfc-editor.org/info/rfc8783>>.
- [RFC8803] Bonaventure, O., Ed., Boucadair, M., Ed., Gundavelli, S., Seo, S., and B. Hesmans, "0-RTT TCP Convert Protocol", RFC 8803, DOI 10.17487/RFC8803, July 2020, <<https://www.rfc-editor.org/info/rfc8803>>.
- [RFC8903] Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use Cases for DDoS Open Threat Signaling", RFC 8903, DOI 10.17487/RFC8903, May 2021, <<https://www.rfc-editor.org/info/rfc8903>>.
- [RFC8973] Boucadair, M. and T. Reddy.K, "DDoS Open Threat Signaling (DOTS) Agent Discovery", RFC 8973, DOI 10.17487/RFC8973, January 2021, <<https://www.rfc-editor.org/info/rfc8973>>.
- [RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <<https://www.rfc-editor.org/info/rfc9132>>.
- [TS.24008] 3GPP, "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3 (Release 16)", December 2019, <<http://www.3gpp.org/DynaReport/24008.htm>>.

Authors' Addresses

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy.K
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India
Email: kondtir@gmail.com

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

DOTS
Internet-Draft
Intended status: Standards Track
Expires: 22 September 2022

M. Boucadair, Ed.
Orange
T. Reddy.K, Ed.
Akamai
E. Doron
Radware Ltd.
M. Chen
CMCC
J. Shallow
21 March 2022

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry
draft-ietf-dots-telemetry-25

Abstract

This document aims to enrich the DOTS signal channel protocol with various telemetry attributes, allowing for optimal Distributed Denial-of-Service (DDoS) attack mitigation. It 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.

This document specifies a YANG module for representing DOTS telemetry message types. It also specifies a second YANG module to share the attack mapping details over the DOTS data channel.

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 22 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Terminology	6
3. DOTS Telemetry: Overview and Purpose	7
3.1. Need More Visibility	7
3.2. Enhanced Detection	8
3.3. Efficient Mitigation	10
4. Design Overview	10
4.1. Overview of Telemetry Operations	11
4.2. Block-wise Transfer	12
4.3. DOTS Multi-homing Considerations	13
4.4. YANG Considerations	13
5. Generic Considerations	14
5.1. DOTS Client Identification	14
5.2. DOTS Gateways	15
5.3. Empty URI Paths	15
5.4. Controlling Configuration Data	15
5.5. Message Validation	15
5.6. A Note About Examples	15
6. Telemetry Operation Paths	16
7. DOTS Telemetry Setup Configuration	17
7.1. Telemetry Configuration	17
7.1.1. Retrieve Current DOTS Telemetry Configuration	18
7.1.2. Conveying DOTS Telemetry Configuration	20
7.1.3. Retrieve Installed DOTS Telemetry Configuration	24
7.1.4. Delete DOTS Telemetry Configuration	25
7.2. Total Pipe Capacity	25
7.2.1. Conveying DOTS Client Domain Pipe Capacity	26
7.2.2. Retrieve Installed DOTS Client Domain Pipe Capacity	32
7.2.3. Delete Installed DOTS Client Domain Pipe Capacity	32
7.3. Telemetry Baseline	32
7.3.1. Conveying DOTS Client Domain Baseline Information	35

7.3.2.	Retrieve Installed Normal Traffic Baseline	39
7.3.3.	Delete Installed Normal Traffic Baseline	39
7.4.	Reset Installed Telemetry Setup	39
7.5.	Conflict with Other DOTS Clients of the Same Domain . . .	39
8.	DOTS Pre-or-Ongoing Mitigation Telemetry	40
8.1.	Pre-or-Ongoing-Mitigation DOTS Telemetry Attributes . . .	42
8.1.1.	Target	43
8.1.2.	Total Traffic	44
8.1.3.	Total Attack Traffic	46
8.1.4.	Total Attack Connections	48
8.1.5.	Attack Details	50
8.1.6.	Vendor Attack Mapping	53
8.2.	From DOTS Clients to DOTS Servers	57
8.3.	From DOTS Servers to DOTS Clients	60
9.	DOTS Telemetry Mitigation Status Update	65
9.1.	DOTS Clients to Servers Mitigation Efficacy DOTS Telemetry Attributes	65
9.2.	DOTS Servers to Clients Mitigation Status DOTS Telemetry Attributes	67
10.	Error Handling	71
11.	YANG Modules	72
11.1.	DOTS Signal Channel Telemetry YANG Module	72
11.2.	Vendor Attack Mapping Details YANG Module	102
12.	YANG/JSON Mapping Parameters to CBOR	106
13.	IANA Considerations	109
13.1.	DOTS Signal Channel CBOR Key Values	109
13.2.	DOTS Signal Channel Conflict Cause Codes	111
13.3.	DOTS Signal Telemetry YANG Module	112
14.	Security Considerations	112
14.1.	DOTS Signal Channel Telemetry	113
14.2.	Vendor Attack Mapping	114
15.	Contributors	115
16.	Acknowledgements	115
17.	References	115
17.1.	Normative References	115
17.2.	Informative References	117
	Authors' Addresses	119

1. Introduction

IT organizations and service providers are facing Distributed Denial of Service (DDoS) attacks that fall into two broad categories:

1. 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 prevent various network elements (routers, switches, firewalls, transit links, and so on) from serving legitimate users' traffic.

The main method of such attacks is to send a large volume or high packet per second (pps) of traffic toward the victim's infrastructure. Typically, attack volumes may vary from a few 100 Mbps to 100s of Gbps or even Tbps. Attacks are commonly carried out leveraging botnets and attack reflectors for amplification attacks (Section 3.1 of [RFC4732]) such as NTP (Network Time Protocol), DNS (Domain Name System), SNMP (Simple Network Management Protocol), or SSDP (Simple Service Discovery Protocol).

2. 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 applications with their port numbers open at network edges can be attractive attack targets.

Application layer attacks are considered more complex and harder to categorize, and 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 against. 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 conclusion derived from the aforementioned attack 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 normal behavior of the targeted systems (including normal traffic patterns), as well as the attacker's ongoing and past actions. Even more challenging, retrieving all the analytics needed for detecting these attacks is not simple with the industry's current reporting capabilities.

The DOTS signal channel protocol [RFC9132] 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 [RFC8903].

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 itself. This can be due to a 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 its DOTS server(s). By enabling the DOTS client to share this comprehensive knowledge of an ongoing attack under specific circumstances, the DOTS server can drastically increase its ability to accomplish successful mitigation. While the attack is being handled by the mitigation resources associated with the DOTS server, the DOTS server has 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.

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 clients and servers can benefit from this information by presenting various information in relevant management, reporting, and portal systems.

This document defines DOTS telemetry attributes that can be conveyed by DOTS clients to DOTS servers, and vice versa. The DOTS telemetry attributes are not mandatory attributes of the DOTS signal channel protocol [RFC9132]. When no limitation policy is provided to a DOTS agent, it can signal available telemetry attributes to its peers in order to optimize the overall mitigation service provisioned using DOTS. The aforementioned policy can be, for example, agreed during a service subscription (that is out of scope) to identify a subset of DOTS clients among those deployed in a DOTS client domain that are allowed to send or receive telemetry data.

Also, the document specifies a YANG module (Section 11.2) that augments the DOTS data channel [RFC8783] with attack details information. Sharing such details during 'idle' time is meant to optimize the data exchanged over the DOTS signal channel.

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 the normal traffic baseline, attacks and their mitigation measures, and any related information that may help in enforcing countermeasures. DOTS Telemetry is an optional set of attributes that can be signaled in the DOTS signal channel protocol.

Telemetry Setup Identifier (tsid) is an identifier that is generated by DOTS clients to uniquely identify DOTS telemetry setup configuration data. See Section 7.1.2 for more details.

Telemetry Identifier (tmid) is an identifier that is generated by DOTS clients to uniquely identify DOTS telemetry data that is communicated prior to or during a mitigation. See Section 8.2 for more details.

When two telemetry requests overlap, "overlapped" lower numeric 'tsid' (or 'tmid') refers to the lower 'tsid' (or 'tmid') value of these overlapping requests.

The term "pipe" represents the maximum level of traffic that the DOTS client domain can receive. Whether a "pipe" is mapped to one or a group of network interfaces is deployment-specific. For example, each interconnection link may be considered as a specific pipe if the DOTS server is hosted by each upstream provider, while the aggregate of all links to connect to upstream network providers can be considered by a DOTS client domain as a single pipe when communicating with a DOTS server not hosted by these upstream providers.

The document uses IANA-assigned Enterprise Numbers. These numbers are also known as "Private Enterprise Numbers" and "SMI (Structure of Management Information) Network Management Private Enterprise Codes" [Private-Enterprise-Numbers].

The meaning of the symbols in YANG tree diagrams are defined in [RFC8340] and [RFC8791].

Consistent with the convention set in Section 2 of [RFC8783], the examples in Section 8.1.6 use `"/restconf"` as the discovered RESTCONF API root path. Within these examples, some protocol header lines are split into multiple lines for display purposes only. When a line ends with backslash (`'\'`) as the last character, the line is wrapped for display purposes. It is considered to be joined to the next line by deleting the backslash, the following line break, and the leading whitespace of the next line.

3. DOTS Telemetry: Overview and Purpose

Timely and effective signaling of up-to-date DDoS telemetry to all elements involved in the mitigation process is essential and improves the overall DDoS mitigation service effectiveness. Bidirectional feedback between DOTS agents is required for increased awareness by each party of the attack and mitigation efforts, supporting a superior and highly efficient attack mitigation service.

3.1. Need More Visibility

When signaling a mitigation request, it is most certainly beneficial for DOTS clients to signal to DOTS servers any knowledge regarding ongoing attacks. This can happen in cases where DOTS clients are asking DOTS servers for support in defending against attacks that they have already detected and/or (partially) mitigated.

If attacks are already detected and categorized within a DOTS client domain, the DOTS server, and its associated mitigation services, can proactively benefit from this information and optimize the overall service delivery. It is important to note that DOTS client domains' and DOTS server domains' detection and mitigation approaches can be different, and can potentially result in different results and attack classifications. The DDoS mitigation service treats the ongoing attack details received from DOTS clients as hints and cannot completely rely or trust the attack details conveyed by DOTS clients.

In addition to the DOTS server directly using telemetry data as operational hints, the DOTS server security operation team also benefits from telemetry data. A basic requirement of security operation teams is to be aware of and get visibility into the attacks they need to handle. This holds especially for the case of ongoing attacks, where DOTS telemetry provides data about the current attack status. Even if some mitigation can be automated, operational teams can use the DOTS telemetry information to be prepared for attack mitigation and to assign the correct resources (operation staff, networking and mitigation) for the specific service. Similarly, security operations personnel at the DOTS client side ask for feedback about their requests for protection. Therefore, it is valuable for DOTS servers to share DOTS telemetry with DOTS clients.

Mutual sharing of information is thus crucial for "closing the mitigation loop" between DOTS clients and servers. For the server side team, it is important to confirm that the same attacks that the DOTS server's mitigation resources are seeing are those that a DOTS client is asking for mitigation of. 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 of them". Cases of inconsistency in attack classification between DOTS clients and servers can be highlighted, and maybe handled, using the DOTS telemetry attributes.

In addition, management and orchestration systems, at both DOTS client and server sides, can use DOTS telemetry as feedback to automate various control and management activities derived from signaled telemetry information.

If the DOTS server's mitigation resources have the capabilities to facilitate the DOTS telemetry, the DOTS server adapts its protection strategy and activates the required countermeasures immediately (automation enabled) for the sake of optimized attack mitigation decisions and actions. The interface from the DOTS server to the mitigator to signal the telemetry data is out of scope.

3.2. Enhanced Detection

DOTS telemetry can also be used as input for determining what values to use for the tuning parameters available on the mitigation resources. 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 predefined threshold for a certain period of time is considered as an attack) to an "anomaly detection" approach. For the latter, it is required to maintain rigorous learning of "normal" behavior, and an "anomaly" (or

an attack) is identified and categorized based on the knowledge about the normal behavior and a deviation from this normal behavior. Statistical and artificial intelligence algorithms (e.g., machine learning) are used such that the actual traffic thresholds are automatically calculated by learning the protected entity's normal traffic behavior during 'idle' time (i.e., no mitigation is active). 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 pattern.

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. For example, a packet 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" that 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 such information to the DOTS server along with the attack traffic levels provides value. 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. Fed with this information, the overall mitigation performances is expected to be improved in terms of time to mitigate, accuracy, and false-negative and false-positive rates.

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 of [RFC8811]). Given that DOTS clients can be integrated in a highly diverse set of scenarios and use cases, this emphasizes the need for knowledge of each DOTS client domain behavior, especially given that common global thresholds for attack detection practically cannot be realized. Each DOTS client domain 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 DOTS server's mitigators to calculate the normal baseline in cases where they do not have any knowledge of the traffic beforehand.

Of course, this information can be provided using out-of-band mechanisms or manual configuration at the risk of unmaintained information becoming inaccurate 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.

3.3. Efficient Mitigation

During a high volume attack, DOTS client pipes can be totally saturated. DOTS clients ask their DOTS servers 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 large volumes of "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 DOTS client domain.

In this case, it can be valuable to DOTS clients to signal to DOTS servers the total pipe capacity, which is the level of traffic the DOTS client domain can absorb from its upstream network. This usually is the circuit size which includes all the packet overheads. Dynamic updates of the condition of pipes between DOTS agents while they are under a DDoS attack is essential (e.g., where multiple DOTS clients share the same physical connectivity pipes). The DOTS server should activate other mechanisms to ensure it does not allow the DOTS client domain's pipes to be saturated unintentionally. The rate-limit action defined in [RFC8783] is a reasonable candidate to achieve this objective; the DOTS client can indicate the type(s) 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 [RFC9133] even when the pipe is overwhelmed.

4. Design Overview

4.1. Overview of Telemetry Operations

The DOTS protocol suite is divided into two logical channels: the signal channel [RFC9132] and data channel [RFC8783]. This division is due to the vastly different requirements placed upon the traffic they carry. The DOTS signal channel must remain available and usable even in the face of attack traffic that might, e.g., saturate one direction of the links involved, rendering acknowledgment-based mechanisms unreliable and strongly incentivizing messages to be small enough to be contained in a single IP packet (Section 2.2 of [RFC8612]). In contrast, the DOTS data channel is available for high-bandwidth data transfer before or after an attack, using more conventional transport protocol techniques (Section 2.3 of [RFC8612]). It is generally preferable to perform advance configuration over the DOTS data channel, including configuring aliases for static or nearly static data sets such as sets of network addresses/prefixes that might be subject to related attacks. This design helps to optimize the use of the DOTS signal channel for the small messages that are important to deliver during an attack. As a reminder, both DOTS signal and data channels require secure communication channels (Section 11 of [RFC9132] and Section 10 of [RFC8783]).

Telemetry information has aspects that correspond to both operational modes (i.e., signal and data channels): there is certainly a need to convey updated information about ongoing attack traffic and targets during an attack, so as to convey detailed information about mitigation status and inform updates to mitigation strategy in the face of adaptive attacks. However, it is also useful to provide mitigation services with a picture of normal or "baseline" traffic towards potential targets to aid in detecting when incoming traffic deviates from normal into being an attack. Also, one might populate a "database" of classifications of known types of attack so that a short attack identifier can be used during attack time to describe an observed attack. This specification does make provision for use of the DOTS data channel for the latter function (Section 8.1.6), but otherwise retains most telemetry functionality in the DOTS signal channel.

Note that it is a functional requirement to convey information about ongoing attack traffic during an attack, and information about baseline traffic uses an essentially identical data structure that is naturally defined to sit next to the description of attack traffic. The related telemetry setup information used to parameterize actual traffic data is also sent over the signal channel, out of expediency.

This document specifies an extension to the DOTS signal channel protocol. Considerations about how to establish, maintain, and make use of the DOTS signal channel are specified in [RFC9132].

Once the DOTS signal channel is established, DOTS clients that support the DOTS telemetry extension proceed with the telemetry setup configuration (e.g., measurement interval, telemetry notification interval, pipe capacity, normal traffic baseline) as detailed in Section 7. DOTS agents can then include DOTS telemetry attributes using the DOTS signal channel (Section 8.1). A DOTS client can use separate messages to share with its DOTS server(s) a set of telemetry data bound to an ongoing mitigation (Section 8.2). Also, a DOTS client that is interested in receiving telemetry notifications related to some of its resources follows the procedure defined in Section 8.3. The DOTS client can then decide to send a mitigation request if the notified attack cannot be mitigated locally within the DOTS client domain.

Aggregate DOTS telemetry data can also be included in efficacy update (Section 9.1) or mitigation update (Section 9.2) messages.

4.2. Block-wise Transfer

DOTS clients can use block wise transfer [RFC7959] with the recommendation detailed in Section 4.4.2 of [RFC9132] 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 CoAP Block1 Option in a PUT request (Section 2.5 of [RFC7959]) to initiate large transfers, but these Block1 transfers are likely to fail if the inbound "pipe" is running full because the transfer requires a message from the server for each block, which would likely be lost in the incoming flood. 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.

Q-Block1 and Q-Block2 Options that are similar to the CoAP Block1 and Block2 Options, but enable robust transmissions of big blocks of data with less packet interchanges using NON messages, are defined in [I-D.ietf-core-new-block]. DOTS implementations can consider the use of Q-Block1 and Q-Block2 Options [I-D.ietf-dots-robust-blocks].

4.3. DOTS Multi-homing Considerations

Considerations for multi-homed DOTS clients to select which DOTS server to contact and which IP prefixes to include in a telemetry message to a given peer DOTS server are discussed in [I-D.ietf-dots-multihoming]. For example, if each upstream network exposes a DOTS server and the DOTS client maintains DOTS channels with all of them, only the information related to prefixes assigned by an upstream network to the DOTS client domain will be signaled via the DOTS channel established with the DOTS server of that upstream network.

Considerations related to whether (and how) a DOTS client gleans some telemetry information (e.g., attack details) it receives from a first DOTS server and share it with a second DOTS server are implementation and deployment specific.

4.4. YANG Considerations

Telemetry messages exchanged between DOTS agents are serialized using Concise Binary Object Representation (CBOR) [RFC8949]. CBOR-encoded payloads are used to carry signal-channel-specific payload messages which convey request parameters and response information such as errors.

This document specifies a YANG module [RFC7950] for representing DOTS telemetry message types (Section 11.1). All parameters in the payload of the DOTS signal channel are mapped to CBOR types as specified in Section 12. As a reminder, Section 3 of [RFC9132] defines the rules for mapping YANG-modeled data to CBOR.

The DOTS telemetry module (Section 11.1) is not intended to be used via NETCONF/RESTCONF for DOTS server management purposes. It serves only to provide a data model and encoding following [RFC8791]. Server deviations (Section 5.6.3 of [RFC7950]) are strongly discouraged, as the peer DOTS agent does not have means to retrieve the list of deviations and thus interoperability issues are likely to be encountered.

The DOTS telemetry module (Section 11.1) uses "enumerations" rather than "identities" to define units, samples, and intervals because otherwise the namespace identifier "ietf-dots-telemetry" must be included when a telemetry attribute is included (e.g., in a mitigation efficacy update). The use of "identities" is thus suboptimal from a message compactness standpoint; one of the key requirements for DOTS Signal Channel messages.

The DOTS telemetry module (Section 11.1) includes some lists for which no key statement is included. This behavior is compliant with [RFC8791]. The reason for not including these keys is because they are not included in the message body of DOTS requests; such keys are included as mandatory Uri-Paths in requests (Sections 7 and 8). Otherwise, whenever a key statement is used in the module, the same definition as in Section 7.8.2 of [RFC7950] is assumed.

Some parameters (e.g., low percentile values) may be associated with different YANG types (e.g., decimal64 and yang:gauge64). To easily distinguish the types of these parameters while using meaningful names, the following suffixes are used:

Suffix	YANG Type	Example
-g	yang:gauge64	low-percentile-g
-c	container	connection-c
-ps	per second	connection-ps

Table 1

The full tree diagram of the DOTS telemetry module can be generated using the "pyang" tool [PYANG]. That tree is not included here because it is too long (Section 3.3 of [RFC8340]). Instead, subtrees are provided for the reader's convenience.

In order to optimize the data exchanged over the DOTS signal channel, the document specifies a second YANG module ("ietf-dots-mapping", Section 11.2) that augments the DOTS data channel [RFC8783]. This augmentation can be used during 'idle' time to share the attack mapping details (Section 8.1.5). DOTS clients can use tools, e.g., YANG Library [RFC8525], to retrieve the list of features and deviations supported by the DOTS server over the data channel.

5. Generic Considerations

5.1. DOTS Client Identification

Following the rules in Section 4.4.1 of [RFC9132], a unique identifier is generated by a DOTS client to prevent request collisions ('cuid').

As a reminder, [RFC9132] forbids 'cuid' to be returned in a response message body.

5.2. DOTS Gateways

DOTS gateways may be located between DOTS clients and servers. The considerations elaborated in Section 4.4.1 of [RFC9132] must be followed. In particular, 'cdid' attribute is used to unambiguously identify a DOTS client domain.

As a reminder, Section 4.4.1.3 of [RFC9132] forbids 'cdid' (if present) to be returned in a response message body.

5.3. Empty URI Paths

Uri-Path parameters and attributes with empty values MUST NOT be present in a request. The presence of such an empty value renders the entire containing message invalid.

5.4. Controlling Configuration Data

The DOTS server follows the same considerations discussed in Section of 4.5.3 of [RFC9132] 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 [RFC9132]. These considerations are not reiterated in the following sections.

5.5. Message Validation

The authoritative reference for validating telemetry messages exchanged over the DOTS signal channel are Sections 7, 8, and 9 together with the mapping table established in Section 12. The structure of telemetry message bodies is represented as a YANG data structure (Section 11.1).

5.6. A Note About Examples

Examples are provided for illustration purposes. The document does not aim to provide a comprehensive list of message examples.

JSON encoding of YANG-modeled data is used to illustrate the various telemetry operations. To ease readability, parameter names and their JSON types are, thus, used in the examples rather than their CBOR key values and CBOR types following the mappings in Section 12. These conventions are inherited from [RFC9132].

The examples use the Enterprise Number 32473 defined for documentation use [RFC5612].

6. Telemetry Operation Paths

As discussed in Section 4.2 of [RFC9132], 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 2):

Operation	Operation Path	Details
Telemetry Setup	/tm-setup	Section 6
Telemetry	/tm	Section 7

Table 2: DOTS Telemetry Operations

Consequently, the "ietf-dots-telemetry" YANG module defined in Section 11.1 defines data structure to represent new DOTS message types called 'telemetry-setup' and 'telemetry'. The tree structure is shown in Figure 1. More details are provided in Sections 7 and 8 about the exact structure of 'telemetry-setup' and 'telemetry' message types.

```

structure dots-telemetry:
  +-- (telemetry-message-type)?
    +--:(telemetry-setup)
      ...
      +-- telemetry* []
        ...
        +-- (setup-type)?
          +--:(telemetry-config)
            ...
          +--:(pipe)
            ...
          +--:(baseline)
            ...
        ...
    +--:(telemetry)
      ...

```

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

DOTS implementations MUST support the Observe Option [RFC7641] for 'tm' (Section 8).

7. DOTS Telemetry Setup Configuration

In reference to Figure 1, a DOTS telemetry setup message MUST include only telemetry-related configuration parameters (Section 7.1) or information about DOTS client domain pipe capacity (Section 7.2) or telemetry traffic baseline (Section 7.3). As such, requests that include a mix of telemetry configuration, pipe capacity, and 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 7.4.

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 7.5.

Telemetry setup configuration is bound to a DOTS client domain. DOTS servers MUST NOT expect DOTS clients to send regular requests to refresh the telemetry setup configuration. Any available telemetry setup configuration is valid till the DOTS server ceases to service a DOTS client domain. DOTS servers MUST NOT reset 'tsid' because a session failed with a DOTS client. DOTS clients update their telemetry setup configuration upon change of a parameter that may impact attack mitigation.

DOTS telemetry setup configuration request and response messages are marked as Confirmable messages (Section 2.1 of [RFC7252]).

7.1. Telemetry Configuration

DOTS telemetry uses several percentile values to provide a picture of a traffic distribution overall, as opposed to just a single snapshot of observed traffic at a single point in time. Modeling raw traffic flow data as a distribution and describing that distribution entails choosing a measurement period that the distribution will describe, and a number of sampling intervals, or "buckets", within that measurement period. Traffic within each bucket is treated as a single event (i.e., averaged), and then the distribution of buckets is used to describe the distribution of traffic over the measurement period. A distribution can be characterized by statistical measures (e.g., mean, median, and standard deviation), and also by reporting

the value of the distribution at various percentile levels of the data set in question (e.g., "quartiles" that correspond to 25th, 50th, and 75th percentile). More details about percentile values and their computation are found in Section 11.3 of [RFC2330].

DOTS telemetry uses up to three percentile values, plus the overall peak, to characterize traffic distributions. Which percentile thresholds are used for these "low", "medium", and "high" percentile values is configurable. Default values are defined in Section 7.1.2.

A DOTS client can negotiate with its server(s) a set of telemetry configuration parameters to be used for telemetry. Such parameters include:

- * Percentile-related measurement parameters. In particular, 'measurement-interval' defines the period on which percentiles are computed, while 'measurement-sample' defines the time distribution for measuring values that are used to compute percentiles.
- * Measurement units
- * Acceptable percentile values
- * Telemetry notification interval
- * Acceptable Server-originated telemetry

7.1.1. 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' Uri-Path when the request is relayed by a DOTS gateway. An example of such a GET request (without gateway) 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 a request, and assuming no error is encountered when processing the request, the DOTS server replies with a 2.05 (Content) response that conveys the telemetry parameters that are

acceptable by the DOTS server, any pipe information (Section 7.2), and the current baseline information (Section 7.3) maintained by the DOTS server for this DOTS client. The tree structure of the response message body is provided in Figure 3.

DOTS servers that support the capability of sending telemetry information to DOTS clients prior to or during a mitigation (Section 9.2) sets 'server-originated-telemetry' under 'max-config-values' to 'true' ('false' is used otherwise). If 'server-originated-telemetry' is not present in a response, this is equivalent to receiving a response with 'server-originated-telemetry' set to 'false'.

```

structure dots-telemetry:
  +-- (telemetry-message-type)?
  +--:(telemetry-setup)
    +-- (direction)?
      +--:(server-to-client-only)
        +-- max-config-values
          +-- measurement-interval?      interval
          +-- measurement-sample?        sample
          +-- low-percentile?             percentile
          +-- mid-percentile?             percentile
          +-- high-percentile?            percentile
          +-- server-originated-telemetry? boolean
          +-- telemetry-notify-interval? uint16
        +-- min-config-values
          +-- measurement-interval?      interval
          +-- measurement-sample?        sample
          +-- low-percentile?             percentile
          +-- mid-percentile?             percentile
          +-- high-percentile?            percentile
          +-- telemetry-notify-interval? uint16
        +-- supported-unit-classes
          +-- unit-config* [unit]
            +-- unit          unit-class
            +-- unit-status   boolean
          +-- supported-query-type* query-type
      +-- telemetry* []
        +-- (direction)?
          +--:(server-to-client-only)
            +-- tsid?          uint32
        +-- (setup-type)?
          +--:(telemetry-config)
            +-- current-config
              +-- measurement-interval?      interval
              +-- measurement-sample?        sample
              +-- low-percentile?             percentile

```

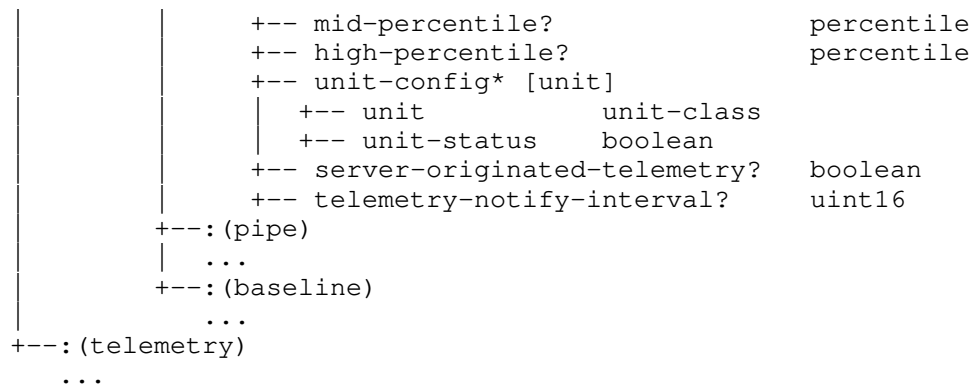


Figure 3: Telemetry Configuration Tree Structure

When both 'min-config-values' and 'max-config-values' attributes are present, the values carried in 'max-config-values' attributes MUST be greater or equal to their counterpart in 'min-config-values' attributes.

7.1.2. Conveying DOTS Telemetry Configuration

A 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 a PUT request. An example of a DOTS client that modifies all percentile reference values is shown in Figure 4.

Note: The payload of the message depicted in Figure 4 is CBOR-encoded as indicated by the Content-Format set to "application/dots+cbor" (Section 10.3 of [RFC9132]). However, and for the sake of better readability, the example (and other similar figures depicting a DOTS telemetry message body) follows the conventions set in Section 5.6: use the JSON names and types defined in Section 12.

```
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-telemetry: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,
depicted as per Section 5.6

'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 configuration data represented as an integer. This identifier MUST be generated by DOTS clients. 'tsid' values MUST increase monotonically whenever new configuration parameters (not just for changed values) need to be conveyed by the DOTS client.

The procedure specified in Section 4.4.1 of [RFC9132] for 'mid' rollover MUST also be followed for 'tsid' rollover.

This is a mandatory attribute. 'tsid' MUST appear after 'cuid' in the Uri-Path options.

'cuid' and 'tsid' MUST NOT appear in the PUT request message body.

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

A 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:

- * 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.
- * 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 2.01 (Created) Response Code MUST be returned in the response.
- * 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.
- * If any of the enclosed configurable attribute values are not acceptable to the DOTS server (Section 7.1.1), 4.22 (Unprocessable Entity) MUST be returned in the response.

The DOTS client may retry and send the PUT request with updated attribute values acceptable to the DOTS server.

By default, low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile), and peak (100th percentile) values are used to represent telemetry data. Nevertheless, a DOTS client can disable some percentile types (low, mid, high). In particular, 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=124"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry: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, depicted as per Section 5.6

DOTS clients can also configure the unit class(es) to be used for traffic-related telemetry data among the following supported unit classes: packets per second, bits per second, and bytes per second. Supplying both bits per second and bytes per second unit-classes is allowed for a given telemetry data. However, receipt of conflicting values is treated as invalid parameters and rejected with 4.00 (Bad Request).

DOTS clients that are interested to receive pre or ongoing mitigation telemetry (pre-or-ongoing-mitigation) information from a DOTS server (Section 9.2) MUST set 'server-originated-telemetry' to 'true'. If 'server-originated-telemetry' is not present in a PUT request, this is equivalent to receiving a request with 'server-originated-telemetry' set to 'false'. An example of a request to enable pre-or-ongoing-mitigation telemetry from DOTS servers 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=125"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "current-config": {
          "server-originated-telemetry": true
        }
      }
    ]
  }
}
```

Figure 6: PUT to Enable Pre-or-ongoing-mitigation Telemetry from the DOTS server, depicted as per Section 5.6

7.1.3. 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 a 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.

7.1.4. 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

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.

Section 7.4 discusses the procedure to reset all DOTS telemetry setup configuration.

7.2. Total Pipe Capacity

A DOTS client can communicate to the DOTS server(s) its DOTS client domain pipe information. The tree structure of the pipe information is shown in Figure 9.

```

structure dots-telemetry:
  +-- (telemetry-message-type)?
  +--:(telemetry-setup)
  |   ...
  |   +-- telemetry* []
  |   |   +-- (direction)?
  |   |   |   +--:(server-to-client-only)
  |   |   |   |   +-- tsid?                               uint32
  |   |   +-- (setup-type)?
  |   |   |   +--:(telemetry-config)
  |   |   |   |   ...
  |   |   |   +--:(pipe)
  |   |   |   |   +-- total-pipe-capacity* [link-id unit]
  |   |   |   |   |   +-- link-id      nt:link-id
  |   |   |   |   |   +-- capacity     uint64
  |   |   |   |   |   +-- unit         unit
  |   |   |   +--:(baseline)
  |   |   |   |   ...
  |   +--:(telemetry)
  |   ...

```

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 of a DOTS client domain. Each of these links is identified with a 'link-id' [RFC8345].

The unit used by a DOTS client when conveying pipe information is captured in the 'unit' attribute. The DOTS client MUST auto-scale so that the appropriate unit is used. That is, for a given unit class, the DOTS client uses the largest unit that gives a value greater than one. As such, only one unit per unit class is allowed.

7.2.1. Conveying DOTS Client Domain Pipe Capacity

Similar considerations to those specified in Section 7.1.2 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 'tsid's used for pipe information. In order to avoid maintaining a long list of 'tsid's 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.

* Note: This assumes that all link information can fit in one single message.

As an example of configuring pipe information, 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 connect to its ISP.

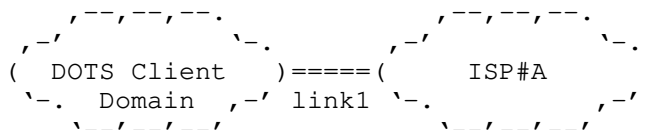


Figure 10: Single Homed DOTS Client Domain

```

Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=126"
Content-Format: "application/dots+cbor"
  
```

```

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "total-pipe-capacity": [
          {
            "link-id": "link1",
            "capacity": "500",
            "unit": "megabit-ps"
          }
        ]
      }
    ]
  }
}
  
```

Figure 11: Example of a PUT Request to Convey Pipe Information (Single Homed), depicted as per Section 5.6

DOTS clients may be instructed to signal a link aggregate instead of individual links. For example, a DOTS client that manages a DOTS client domain having two interconnection links with an upstream ISP (Figure 12) can send a PUT request (shown in Figure 13) to communicate the aggregate link capacity with its ISP. Signaling individual or aggregate link capacity is deployment specific.

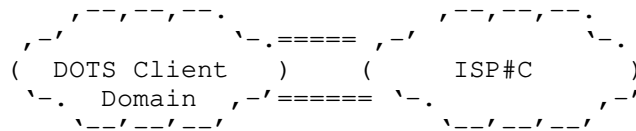


Figure 12: DOTS Client Domain with Two Interconnection Links

```

Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=hmcpH87lmPGsSTjkhXCbin"
Uri-Path: "tsid=896"
Content-Format: "application/dots+cbor"
  
```

```

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "total-pipe-capacity": [
          {
            "link-id": "aggregate",
            "capacity": "700",
            "unit": "megabit-ps"
          }
        ]
      }
    ]
  }
}
  
```

Figure 13: Example of a PUT Request to Convey Pipe Information (Aggregated Link), depicted as per Section 5.6

Now consider that the DOTS client domain was upgraded to connect to an additional ISP (e.g., ISP#B of Figure 14); the DOTS client can inform a DOTS server that is not hosted with ISP#A and ISP#B domains

about this update by sending the PUT request depicted in Figure 15. This request also includes information related to "link1" even if that link is not upgraded. Upon receipt of this request, the DOTS server removes the request with 'tsid=126' and updates its configuration base to maintain two links (link#1 and link#2).

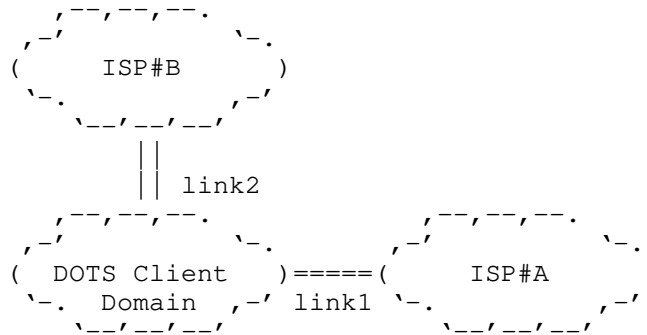


Figure 14: Multi-Homed DOTS Client Domain

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=127"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "total-pipe-capacity": [
          {
            "link-id": "link1",
            "capacity": "500",
            "unit": "megabit-ps"
          },
          {
            "link-id": "link2",
            "capacity": "500",
            "unit": "megabit-ps"
          }
        ]
      }
    ]
  }
}
```

Figure 15: Example of a PUT Request to Convey Pipe Information (Multi-Homed), depicted as per Section 5.6

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 7.2.3 for other delete cases). For example, if a DOTS client domain re-homes (that is, it changes its ISP), the DOTS client can inform its DOTS server about this update (e.g., from the network configuration in Figure 10 to the one shown in Figure 16) by sending the PUT request depicted in Figure 17. Upon receipt of this request, and assuming no error is encountered when processing the request, the DOTS server removes "link1" from its configuration bases for this DOTS client domain. Note that if the DOTS server receives a PUT request with a 'capacity' attribute set to "0" for all included links, it MUST reject the request with a 4.00 (Bad Request). Instead, the DOTS client can use a DELETE request to delete all links (Section 7.2.3).

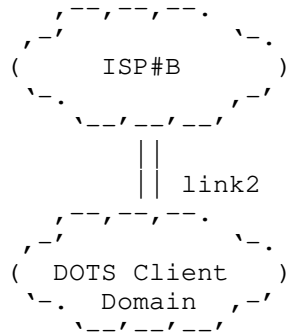


Figure 16: Multi-Homed DOTS Client Domain

```

Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=128"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "total-pipe-capacity": [
          {
            "link-id": "link1",
            "capacity": "0",
            "unit": "megabit-ps"
          },
          {
            "link-id": "link2",
            "capacity": "500",
            "unit": "megabit-ps"
          }
        ]
      }
    ]
  }
}

```

Figure 17: Example of a PUT Request to Convey Pipe Information (Multi-Homed), depicted as per Section 5.6

7.2.2. Retrieve Installed 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 7.1.3 is followed.

To retrieve all pipe information bound to a DOTS client, the DOTS client proceeds as specified in Section 7.1.1.

7.2.3. Delete Installed 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 7.1.4 is followed.

7.3. Telemetry Baseline

A DOTS client can communicate to its DOTS server(s) its normal traffic baseline and connections capacity:

Total traffic normal baseline: The percentile values representing the total traffic normal baseline. It can be represented for a target using 'total-traffic-normal'.

The traffic normal per-protocol ('total-traffic-normal-per-protocol') baseline is represented for a target and is transport-protocol specific.

The traffic normal per-port-number ('total-traffic-normal-per-port') baseline is represented for each port number bound to a target.

If the DOTS client negotiated percentile values and units (Section 7.1), these negotiated parameters will be used instead of the default ones. For each used unit class, the DOTS client MUST auto-scale so that the appropriate unit is used.

Total connections capacity: If the target is susceptible 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 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. Embryonic connection is only possible in connection-oriented transport protocols like TCP or Stream Control Transmission Protocol (SCTP) [RFC4960].
- * 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 (e.g., HTTP/DNS/SIP requests) allowed per second to the target.
- * The maximum number of requests allowed per second to the target per client.
- * The maximum number of outstanding partial requests allowed to the target. Attacks relying upon partial requests create a connection with a target but do not send a complete request (e.g., HTTP request).
- * The maximum number of outstanding partial requests allowed to the target per client.

The aggregate per transport protocol is captured in 'total-connection-capacity', while port-specific capabilities are represented using 'total-connection-capacity-per-port'.

Note that a target resource is identified using the attributes 'target-prefix', 'target-port-range', 'target-protocol', 'target-fqdn', 'target-uri', or 'alias-name' defined in Section 4.4.1.1 of [RFC9132].

The tree structure of the normal traffic baseline is shown in Figure 18.

```
structure dots-telemetry:
  +-- (telemetry-message-type)?
    +--:(telemetry-setup)
      |
      | ...
      | +-- telemetry* []
      |   +-- (direction)?
```

```

|   +---:(server-to-client-only)
|       +--- tsid?                               uint32
+--- (setup-type)?
|   +---:(telemetry-config)
|       ...
+---:(pipe)
|   ...
+---:(baseline)
|   +--- baseline* [id]
|       +--- id
|           uint32
|       +--- target-prefix*
|           inet:ip-prefix
|       +--- target-port-range* [lower-port]
|           +--- lower-port    inet:port-number
|           +--- upper-port?   inet:port-number
|       +--- target-protocol*                               uint8
|       +--- target-fqdn*
|           inet:domain-name
|       +--- target-uri*
|           inet:uri
|       +--- alias-name*
|           string
|       +--- total-traffic-normal* [unit]
|           +--- unit                unit
|           +--- low-percentile-g?   yang:gauge64
|           +--- mid-percentile-g?   yang:gauge64
|           +--- high-percentile-g?  yang:gauge64
|           +--- peak-g?             yang:gauge64
|       +--- total-traffic-normal-per-protocol*
|           [unit protocol]
|           +--- protocol            uint8
|           +--- unit                unit
|           +--- low-percentile-g?   yang:gauge64
|           +--- mid-percentile-g?   yang:gauge64
|           +--- high-percentile-g?  yang:gauge64
|           +--- peak-g?             yang:gauge64
|       +--- total-traffic-normal-per-port* [unit port]
|           +--- port                inet:port-number
|           +--- unit                unit
|           +--- low-percentile-g?   yang:gauge64
|           +--- mid-percentile-g?   yang:gauge64
|           +--- high-percentile-g?  yang:gauge64
|           +--- peak-g?             yang:gauge64
|       +--- total-connection-capacity* [protocol]
|           +--- protocol            uint8
|           +--- connection?         uint64
|           +--- connection-client?  uint64

```

```

+--- embryonic?                               uint64
+--- embryonic-client?                         uint64
+--- connection-ps?                           uint64
+--- connection-client-ps?                    uint64
+--- request-ps?                              uint64
+--- request-client-ps?                       uint64
+--- partial-request-max?                     uint64
+--- partial-request-client-max?              uint64
+--- total-connection-capacity-per-port*
    [protocol port]
    +--- port
    |   inet:port-number
    +--- protocol                             uint8
+--- connection?                             uint64
+--- connection-client?                       uint64
+--- embryonic?                               uint64
+--- embryonic-client?                       uint64
+--- connection-ps?                           uint64
+--- connection-client-ps?                    uint64
+--- request-ps?                              uint64
+--- request-client-ps?                       uint64
+--- partial-request-max?                     uint64
+--- partial-request-client-max?              uint64
+---:(telemetry)
...

```

Figure 18: Telemetry Baseline Tree Structure

A DOTS client can share one or multiple normal traffic baselines (e.g., aggregate or per-prefix baselines), each are uniquely identified within the DOTS client domain with an identifier 'id'. This identifier can be used to update a baseline entry, delete a specific entry, etc.

7.3.1. Conveying DOTS Client Domain Baseline Information

Similar considerations to those specified in Section 7.1.2 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, URI, or alias-name. Also, two PUT requests from a DOTS client have overlapping targets from the perspective of the DOTS server if the addresses associated with the FQDN, URI, or alias are overlapping with each other or with 'target-prefix'.

DOTS clients SHOULD minimize the number of active 'tsid's used for baseline information. In order to avoid maintaining a long list of 'tsid's 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 attribute is 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 19.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=129"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "baseline": [
          {
            "id": 1,
            "target-prefix": [
              "2001:db8:6401::1/128",
              "2001:db8:6401::2/128"
            ],
            "total-traffic-normal": [
              {
                "unit": "megabit-ps",
                "peak-g": "60"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Figure 19: PUT to Conveying the DOTS Traffic Baseline, depicted as per Section 5.6

The DOTS client may share protocol specific baseline information (e.g., TCP and UDP) as shown in Figure 20.

```

Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=130"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "baseline": [
          {
            "id": 1,
            "target-prefix": [
              "2001:db8:6401::1/128",
              "2001:db8:6401::2/128"
            ],
            "total-traffic-normal-per-protocol": [
              {
                "unit": "megabit-ps",
                "protocol": 6,
                "peak-g": "50"
              },
              {
                "unit": "megabit-ps",
                "protocol": 17,
                "peak-g": "10"
              }
            ]
          }
        ]
      }
    ]
  }
}

```

Figure 20: PUT to Convey the DOTS Traffic Baseline (2), depicted as per Section 5.6

The normal traffic baseline information should be updated to reflect legitimate overloads (e.g., flash crowds) to prevent unnecessary mitigation.

7.3.2. Retrieve Installed 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 7.1.3 is followed.

To retrieve all baseline information bound to a DOTS client, the DOTS client proceeds as specified in Section 7.1.1.

7.3.3. Delete Installed 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 7.1.4 is followed.

7.4. Reset Installed Telemetry Setup

Upon bootstrapping (or reboot or any other event that may alter the DOTS client setup), 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 a request is depicted in Figure 21.

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 21: Delete Telemetry Configuration

7.5. Conflict with Other DOTS Clients of the Same Domain

A DOTS server may detect conflicts between requests conveying 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 [RFC9132]. The conflict cause can be set to one of these values:

1: Overlapping targets (Section 4.4.1 of [RFC9132]).

TBA: Overlapping pipe scope (see Section 13).

8. DOTS Pre-or-Ongoing Mitigation Telemetry

There are two broad types of DDoS attacks: one is a bandwidth consuming attack, the other is a target-resource-consuming attack. This section outlines the set of DOTS telemetry attributes (Section 8.1) that covers both types of attack. The objective of these attributes is to allow for the complete knowledge of attacks and the various particulars that can best characterize attacks.

The "ietf-dots-telemetry" YANG module (Section 11.1) defines the data structure of a new message type called 'telemetry'. The tree structure of the 'telemetry' message type is shown in Figure 24.

The pre-or-ongoing-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. Pre-or-ongoing-mitigation telemetry attributes specified in Section 8.1 can be signaled between DOTS agents.

Pre-or-ongoing-mitigation telemetry attributes may be sent by a DOTS client or a DOTS server.

DOTS agents SHOULD bind pre-or-ongoing-mitigation telemetry data to mitigation requests associated with the resources under attack. In particular, a telemetry PUT request sent after a mitigation request may include a reference to that mitigation request ('mid-list') as shown in Figure 22. An example illustrating request correlation by means of 'target-prefix' is shown in Figure 23.

Many of the pre-or-ongoing-mitigation telemetry data use a unit that falls under the unit class that is configured following the procedure described in Section 7.1.2. When generating telemetry data to send to a peer, the DOTS agent MUST auto-scale so that appropriate unit(s) are used.

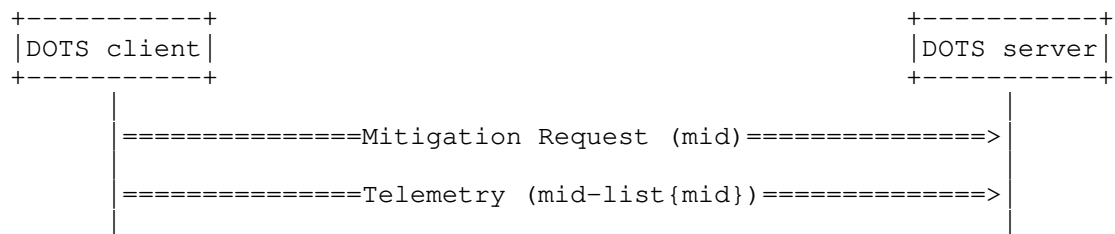


Figure 22: Example of Request Correlation using 'mid'

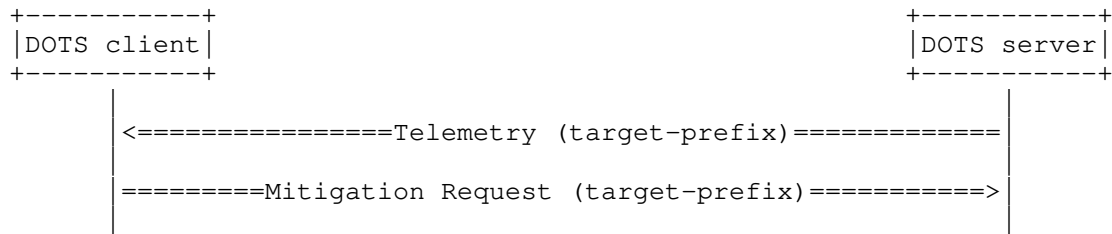


Figure 23: Example of Request Correlation using Target Prefix

DOTS agents MUST NOT send pre-or-ongoing-mitigation telemetry notifications to the same peer more frequently than once every 'telemetry-notify-interval' (Section 7.1). If a telemetry notification is sent using a block-like transfer mechanism (e.g., [I-D.ietf-core-new-block]), this rate limit policy MUST NOT consider these individual blocks as separate notifications, but as a single notification.

DOTS pre-or-ongoing-mitigation telemetry request and response messages MUST be marked as Non-Confirmable messages (Section 2.1 of [RFC7252]).

```

structure dots-telemetry:
  +-- (telemetry-message-type)?
  +--:(telemetry-setup)
  |   ...
  |   +-- telemetry* []
  |   |   +-- (direction)?
  |   |   |   +--:(server-to-client-only)
  |   |   |   |   +-- tsid?                               uint32
  |   |   +-- (setup-type)?
  |   |   |   +--:(telemetry-config)
  |   |   |   |   ...
  |   |   |   +--:(pipe)
  |   |   |   |   ...
  |   |   |   +--:(baseline)
  |   |   |   |   ...
  |   +--:(telemetry)
  |   |   +-- pre-or-ongoing-mitigation* []
  |   |   |   +-- (direction)?
  |   |   |   |   +--:(server-to-client-only)
  |   |   |   |   |   +-- tmid?                               uint32
  |   |   |   +-- target
  |   |   |   |   ...
  |   |   |   +-- total-traffic* [unit]
  |   |   |   |   ...
  |   |   |   +-- total-traffic-protocol* [unit protocol]
  |   |   |   |   ...
  |   |   |   +-- total-traffic-port* [unit port]
  |   |   |   |   ...
  |   |   |   +-- total-attack-traffic* [unit]
  |   |   |   |   ...
  |   |   |   +-- total-attack-traffic-protocol* [unit protocol]
  |   |   |   |   ...
  |   |   |   +-- total-attack-traffic-port* [unit port]
  |   |   |   |   ...
  |   |   |   +-- total-attack-connection-protocol* [protocol]
  |   |   |   |   ...
  |   |   |   +-- total-attack-connection-port* [protocol port]
  |   |   |   |   ...
  |   |   |   +-- attack-detail* [vendor-id attack-id]
  |   |   |   |   ...
  |   |   |   ...

```

Figure 24: Telemetry Message Type Tree Structure

8.1. Pre-or-Ongoing-Mitigation DOTS Telemetry Attributes

The description and motivation behind each attribute are presented in Section 3.

8.1.1. Target

A target resource (Figure 25) is identified using the attributes 'target-prefix', 'target-port-range', 'target-protocol', 'target-fqdn', 'target-uri', 'alias-name', or a pointer to a mitigation request ('mid-list').

```

+--:(telemetry)
  +-- pre-or-ongoing-mitigation* []
    +-- (direction)?
      +--:(server-to-client-only)
        +-- tmid?                               uint32
      +-- target
        +-- target-prefix*                       inet:ip-prefix
        +-- target-port-range* [lower-port]
          +-- lower-port                       inet:port-number
          +-- upper-port?                     inet:port-number
        +-- target-protocol*                     uint8
        +-- target-fqdn*                         inet:domain-name
        +-- target-uri*                         inet:uri
        +-- alias-name*                         string
        +-- mid-list*                           uint32
      +-- total-traffic* [unit]
        ...
      +-- total-traffic-protocol* [unit protocol]
        ...
      +-- total-traffic-port* [unit port]
        ...
      +-- total-attack-traffic* [unit]
        ...
      +-- total-attack-traffic-protocol* [unit protocol]
        ...
      +-- total-attack-traffic-port* [unit port]
        ...
      +-- total-attack-connection-protocol* [protocol]
        ...
      +-- total-attack-connection-port* [protocol port]
        ...
      +-- attack-detail* [vendor-id attack-id]
        ...

```

Figure 25: Target Tree Structure

At least one of the attributes 'target-prefix', 'target-fqdn', 'target-uri', 'alias-name', or 'mid-list' MUST be present in the target definition.

If the target is susceptible to bandwidth-consuming attacks, the attributes representing the percentile values of the 'attack-id' attack traffic are included.

If the target is susceptible to resource-consuming DDoS attacks, the attributes defined in Section 8.1.4 are applicable for representing the attack.

At least the 'target' attribute and one other pre-or-ongoing-mitigation attribute MUST be present in the DOTS telemetry message.

8.1.2. Total Traffic

The 'total-traffic' attribute (Figure 26) conveys the percentile values (including peak and current observed values) of the total observed traffic. More fine-grained information about the total traffic can be conveyed in the 'total-traffic-protocol' and 'total-traffic-port' attributes.

The 'total-traffic-protocol' attribute represents the total traffic for a target and is transport-protocol specific.

The 'total-traffic-port' represents the total traffic for a target per port number.

```

+--:(telemetry)
  +-- pre-or-ongoing-mitigation* []
    +-- (direction)?
      +--:(server-to-client-only)
        +-- tmid?                               uint32
      +-- target
      | ...
    +-- total-traffic* [unit]
      +-- unit                                   unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                               yang:gauge64
      +-- current-g?                            yang:gauge64
    +-- total-traffic-protocol* [unit protocol]
      +-- protocol                             uint8
      +-- unit                                   unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                               yang:gauge64
      +-- current-g?                            yang:gauge64
    +-- total-traffic-port* [unit port]
      +-- port                                 inet:port-number
      +-- unit                                   unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                               yang:gauge64
      +-- current-g?                            yang:gauge64
    +-- total-attack-traffic* [unit]
      | ...
    +-- total-attack-traffic-protocol* [unit protocol]
      | ...
    +-- total-attack-traffic-port* [unit port]
      | ...
    +-- total-attack-connection-protocol* [protocol]
      | ...
    +-- total-attack-connection-port* [protocol port]
      | ...
    +-- attack-detail* [vendor-id attack-id]
      | ...

```

Figure 26: Total Traffic Tree Structure

8.1.3. Total Attack Traffic

The 'total-attack-traffic' attribute (Figure 27) conveys the total observed attack traffic. More fine-grained information about the total attack traffic can be conveyed in the 'total-attack-traffic-protocol' and 'total-attack-traffic-port' attributes.

The 'total-attack-traffic-protocol' attribute represents the total attack traffic for a target and is transport-protocol specific.

The 'total-attack-traffic-port' attribute represents the total attack traffic for a target per port number.


```

+--:(telemetry)
  +-- pre-or-ongoing-mitigation* []
    +-- (direction)?
      +--:(server-to-client-only)
        +-- tmid?                               uint32
      +-- target
      | ...
    +-- total-traffic* [unit]
      | ...
    +-- total-traffic-protocol* [unit protocol]
      | ...
    +-- total-traffic-port* [unit port]
      | ...
    +-- total-attack-traffic* [unit]
      +-- unit                                   unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                              yang:gauge64
      +-- current-g?                           yang:gauge64
    +-- total-attack-traffic-protocol* [unit protocol]
      +-- protocol                             uint8
      +-- unit                                 unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                              yang:gauge64
      +-- current-g?                           yang:gauge64
    +-- total-attack-traffic-port* [unit port]
      +-- port                                 inet:port-number
      +-- unit                                 unit
      +-- low-percentile-g?                     yang:gauge64
      +-- mid-percentile-g?                     yang:gauge64
      +-- high-percentile-g?                    yang:gauge64
      +-- peak-g?                              yang:gauge64
      +-- current-g?                           yang:gauge64
    +-- total-attack-connection-protocol* [protocol]
      | ...
    +-- total-attack-connection-port* [protocol port]
      | ...
    +-- attack-detail* [vendor-id attack-id]
      ...

```

Figure 27: Total Attack Traffic Tree Structure

8.1.4. Total Attack Connections

If the target is susceptible to resource-consuming DDoS attacks, the 'total-attack-connection-protocol' attribute is used to convey the percentile values (including peak and current observed values) of various attributes related to the total attack connections. The following optional sub-attributes for the target per transport protocol are included to represent the attack characteristics:

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

The total attack connections per port number is represented using the 'total-attack-connection-port' attribute.

```

+--:(telemetry)
  +-- pre-or-ongoing-mitigation* []
    +-- (direction)?
      +--:(server-to-client-only)
        +-- tmid?                               uint32
      +-- target
      | ...
      +-- total-traffic* [unit]
      | ...
      +-- total-traffic-protocol* [unit protocol]
      | ...
      +-- total-traffic-port* [unit port]
      | ...
      +-- total-attack-traffic* [unit]
      | ...
      +-- total-attack-traffic-protocol* [unit protocol]
      | ...
      +-- total-attack-traffic-port* [unit port]
      | ...
      +-- total-attack-connection-protocol* [protocol]
        +-- protocol                               uint8
        +-- connection-c
          +-- low-percentile-g?      yang:gauge64
          +-- mid-percentile-g?      yang:gauge64
          +-- high-percentile-g?     yang:gauge64
          +-- peak-g?                yang:gauge64
          +-- current-g?             yang:gauge64
        +-- embryonic-c
          +-- low-percentile-g?      yang:gauge64
          +-- mid-percentile-g?      yang:gauge64

```

```

| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- connection-ps-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- request-ps-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- partial-request-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- total-attack-connection-port* [protocol port]
| | +-- protocol uint8
| | +-- port inet:port-number
+-- connection-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- embryonic-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- connection-ps-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64
+-- request-ps-c
| | +-- low-percentile-g? yang:gauge64
| | +-- mid-percentile-g? yang:gauge64
| | +-- high-percentile-g? yang:gauge64
| | +-- peak-g? yang:gauge64
| | +-- current-g? yang:gauge64

```

```

    +-- partial-request-c
      +-- low-percentile-g?   yang:gauge64
      +-- mid-percentile-g?  yang:gauge64
      +-- high-percentile-g? yang:gauge64
      +-- peak-g?            yang:gauge64
      +-- current-g?         yang:gauge64
    +-- attack-detail* [vendor-id attack-id]
    ...

```

Figure 28: Total Attack Connections Tree Structure

8.1.5. Attack Details

This attribute (depicted in Figure 29) is used to signal a set of details characterizing an attack. The following sub-attributes describing the ongoing attack can be signalled as attack details:

vendor-id: Vendor ID is a security vendor's enterprise number as registered in the IANA's "Private Enterprise Numbers" registry [Private-Enterprise-Numbers].

attack-id: Unique identifier assigned for the attack by a vendor. This parameter **MUST** be present independent of whether 'attack-description' is included or not.

description-lang: Indicates the language tag that is used for the text that is included in the 'attack-description' attribute. The attribute is encoded following the rules in Section 2.1 of [RFC5646]. The default language tag is "en-US".

attack-description: Textual representation of the attack description. This description is related to the class of attack rather than a specific instance of it. Natural Language Processing techniques (e.g., word embedding) might provide some utility in mapping 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 and (b) avoids the need to standardize attack types which keep evolving.

attack-severity: Attack severity level. This attribute takes one of the values defined in Section 3.12.2 of [RFC7970].

start-time: The time the attack started. The attack's start time is expressed in seconds relative to 1970-01-01T00:00Z (Section 3.4.2 of [RFC8949]). The CBOR encoding is modified so that the leading tag 1 (epoch-based date/time) **MUST** be omitted.

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

expressed in seconds relative to 1970-01-01T00:00Z (Section 3.4.2 of [RFC8949]). The CBOR encoding is modified so that the leading tag 1 (epoch-based date/time) MUST be omitted.

source-count: A count of sources involved in the attack targeting the victim.

top-talker: A list of attack sources that are involved in an attack and which are generating an important part of the attack traffic. The top talkers are represented using the 'source-prefix'.

'spoofed-status' indicates whether a top talker is a spoofed IP address (e.g., reflection attacks) or not. If no 'spoofed-status' data node is included, this means that the spoofing status is unknown.

If the target is being subjected to a bandwidth-consuming attack, a statistical profile of the attack traffic from each of the top talkers is included ('total-attack-traffic', Section 8.1.3).

If the target is being subjected to a resource-consuming DDoS attack, the same attributes defined in Section 8.1.4 are applicable for characterizing the attack on a per-talker basis.

```

+--:(telemetry)
  +-- pre-or-ongoing-mitigation* []
    +-- (direction)?
      |   +--:(server-to-client-only)
      |     +-- tmid?                               uint32
    +-- target
      |   ...
    +-- total-traffic* [unit]
      |   ...
    +-- total-traffic-protocol* [unit protocol]
      |   ...
    +-- total-traffic-port* [unit port]
      |   ...
    +-- total-attack-traffic* [unit]
      |   ...
    +-- total-attack-traffic-protocol* [unit protocol]
      |   ...
    +-- total-attack-traffic-port* [unit port]
      |   ...
    +-- total-attack-connection-protocol* [protocol]
      |   ...
    +-- total-attack-connection-port* [protocol port]
      |   ...
    +-- attack-detail* [vendor-id attack-id]

```

```

+-- vendor-id                uint32
+-- attack-id                uint32
+-- description-lang?        string
+-- attack-description?      string
+-- attack-severity?         attack-severity
+-- start-time?              uint64
+-- end-time?                uint64
+-- source-count
|   +-- low-percentile-g?     yang:gauge64
|   +-- mid-percentile-g?     yang:gauge64
|   +-- high-percentile-g?    yang:gauge64
|   +-- peak-g?              yang:gauge64
|   +-- current-g?           yang:gauge64
+-- top-talker
|   +-- talker* [source-prefix]
|   |   +-- spoofed-status?    boolean
|   |   +-- source-prefix      inet:ip-prefix
|   |   +-- source-port-range* [lower-port]
|   |   |   +-- lower-port     inet:port-number
|   |   |   +-- upper-port?    inet:port-number
|   |   +-- source-icmp-type-range* [lower-type]
|   |   |   +-- lower-type     uint8
|   |   |   +-- upper-type?    uint8
|   |   +-- total-attack-traffic* [unit]
|   |   |   +-- unit           unit
|   |   |   +-- low-percentile-g? yang:gauge64
|   |   |   +-- mid-percentile-g? yang:gauge64
|   |   |   +-- high-percentile-g? yang:gauge64
|   |   |   +-- peak-g?        yang:gauge64
|   |   |   +-- current-g?     yang:gauge64
|   |   +-- total-attack-connection-protocol*
|   |   |   [protocol]
|   |   |   +-- protocol      uint8
|   |   +-- connection-c
|   |   |   +-- low-percentile-g? yang:gauge64
|   |   |   +-- mid-percentile-g? yang:gauge64
|   |   |   +-- high-percentile-g? yang:gauge64
|   |   |   +-- peak-g?        yang:gauge64
|   |   |   +-- current-g?     yang:gauge64
|   |   +-- embryonic-c
|   |   |   +-- low-percentile-g? yang:gauge64
|   |   |   +-- mid-percentile-g? yang:gauge64
|   |   |   +-- high-percentile-g? yang:gauge64
|   |   |   +-- peak-g?        yang:gauge64
|   |   |   +-- current-g?     yang:gauge64
|   |   +-- connection-ps-c
|   |   |   +-- low-percentile-g? yang:gauge64
|   |   |   +-- mid-percentile-g? yang:gauge64

```

```

| +-- high-percentile-g?  yang:gauge64
| +-- peak-g?            yang:gauge64
| +-- current-g?         yang:gauge64
+-- request-ps-c
| +-- low-percentile-g?   yang:gauge64
| +-- mid-percentile-g?   yang:gauge64
| +-- high-percentile-g?  yang:gauge64
| +-- peak-g?            yang:gauge64
| +-- current-g?         yang:gauge64
+-- partial-request-c
  +-- low-percentile-g?   yang:gauge64
  +-- mid-percentile-g?   yang:gauge64
  +-- high-percentile-g?  yang:gauge64
  +-- peak-g?            yang:gauge64
  +-- current-g?         yang:gauge64

```

Figure 29: Attack Detail Tree Structure

In order to optimize the size of telemetry data conveyed over the DOTS signal channel, DOTS agents MAY use the DOTS data channel [RFC8783] to exchange vendor specific attack mapping details (that is, {vendor identifier, attack identifier} ==> textual representation of the attack description). As such, DOTS agents do not have to convey systematically an attack description in their telemetry messages over the DOTS signal channel. Refer to Section 8.1.6.

8.1.6. Vendor Attack Mapping

Multiple mappings for different vendor identifiers may be used; the DOTS agent transmitting telemetry information can elect to use one or more vendor mappings even in the same telemetry message.

Note: It is possible that a DOTS server is making use of multiple DOTS mitigators; each from a different vendor. How telemetry information and vendor mappings are exchanged between DOTS servers and DOTS mitigators is outside the scope of this document.

DOTS clients and servers may be provided with mappings from different vendors and so have their own different sets of vendor attack mappings. A DOTS agent MUST accept receipt of telemetry data with a vendor identifier that is different to the one it uses to transmit telemetry data. Furthermore, it is possible that the DOTS client and DOTS server are provided by the same vendor, but the vendor mapping tables are at different revisions. The DOTS client SHOULD transmit telemetry information using any vendor mapping(s) that it provided to the DOTS server (e.g., using a POST as depicted in Figure 34) and the DOTS server SHOULD use any vendor mappings(s) provided to the DOTS client when transmitting telemetry data to the peer DOTS agent.

The "ietf-dots-mapping" YANG module defined in Section 11.2 augments the "ietf-dots-data-channel" [RFC8783] module. The tree structure of the "ietf-dots-mapping" module is shown in Figure 30.

```

module: ietf-dots-mapping
  augment /data-channel:dots-data/data-channel:dots-client:
    +--rw vendor-mapping {dots-telemetry}?
      +--rw vendor* [vendor-id]
        +--rw vendor-id      uint32
        +--rw vendor-name?   string
        +--rw description-lang? string
        +--rw last-updated   uint64
        +--rw attack-mapping* [attack-id]
          +--rw attack-id      uint32
          +--rw attack-description string
  augment /data-channel:dots-data/data-channel:capabilities:
    +--ro vendor-mapping-enabled? boolean {dots-telemetry}?
  augment /data-channel:dots-data:
    +--ro vendor-mapping {dots-telemetry}?
      +--ro vendor* [vendor-id]
        +--ro vendor-id      uint32
        +--ro vendor-name?   string
        +--ro description-lang? string
        +--ro last-updated   uint64
        +--ro attack-mapping* [attack-id]
          +--ro attack-id      uint32
          +--ro attack-description string

```

Figure 30: Vendor Attack Mapping Tree Structure

A DOTS client sends a GET request over the DOTS data channel to retrieve the capabilities supported by a DOTS server as per Section 7.1 of [RFC8783]. This request is meant to assess whether the capability of sharing vendor attack mapping details is supported by the server (i.e., check the value of 'vendor-mapping-enabled').

If 'vendor-mapping-enabled' is set to 'true', a DOTS client MAY send a GET request to retrieve the DOTS server's vendor attack mapping details. An example of such a GET request is shown in Figure 31.

```

GET /restconf/data/ietf-dots-data-channel:dots-data\
  /ietf-dots-mapping:vendor-mapping HTTP/1.1
Host: example.com
Accept: application/yang-data+json

```

Figure 31: GET to Retrieve the Vendor Attack Mappings of a DOTS Server

A DOTS client can retrieve only the list of vendors supported by the DOTS server. It does so by setting the "depth" parameter (Section 4.8.2 of [RFC8040]) to "3" in the GET request as shown in Figure 32. An example of a response body received from the DOTS server as a response to such a request is illustrated in Figure 33.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /ietf-dots-mapping:vendor-mapping?depth=3 HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Figure 32: GET to Retrieve the Vendors List used by a DOTS Server

```
{
  "ietf-dots-mapping:vendor-mapping": {
    "vendor": [
      {
        "vendor-id": 32473,
        "vendor-name": "mitigator-s",
        "last-updated": "1629898758",
        "attack-mapping": []
      }
    ]
  }
}
```

Figure 33: Response Message Body to a GET to Retrieve the Vendors List used by a DOTS Server

The DOTS client repeats the above procedure regularly (e.g., once a week) to update the DOTS server's vendor attack mapping details.

If the DOTS client concludes that the DOTS server does not have any reference to the specific vendor attack mapping details, the DOTS client uses a POST request to install its vendor attack mapping details. An example of such a POST request is depicted in Figure 34.

```
POST /restconf/data/ietf-dots-data-channel:dots-data\
/dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-mapping:vendor-mapping": {
    "vendor": [
      {
        "vendor-id": 345,
        "vendor-name": "mitigator-c",
        "last-updated": "1629898958",
        "attack-mapping": [
          {
            "attack-id": 1,
            "attack-description":
              "Include a description of this attack"
          },
          {
            "attack-id": 2,
            "attack-description":
              "Again, include a description of the attack"
          }
        ]
      }
    ]
  }
}
```

Figure 34: POST to Install Vendor Attack Mapping Details

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

If the request is received via a server-domain DOTS gateway, but the DOTS server does not maintain a 'cdid' for this 'cuid' while a 'cdid' is expected to be supplied, the DOTS server MUST reply with "403 Forbidden" status-line and the error-tag "access-denied". Upon receipt of this message, the DOTS client MUST register (Section 5.1 of [RFC8783]).

The DOTS client uses the PUT request to modify its vendor attack mapping details maintained by the DOTS server (e.g., add a new mapping entry, update an existing mapping).

A DOTS client uses a GET request to retrieve its vendor attack mapping details as maintained by the DOTS server (Figure 35).

```
GET /restconf/data/ietf-dots-data-channel:dots-data\  
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\  
    /ietf-dots-mapping:vendor-mapping?\  
    content=all HTTP/1.1  
Host: example.com  
Accept: application/yang-data+json
```

Figure 35: GET to Retrieve Installed Vendor Attack Mapping Details

When conveying attack details in DOTS telemetry messages (Sections 8.2, 8.3, and 9), DOTS agents MUST NOT include the 'attack-description' attribute unless the corresponding attack mapping details were not previously shared with the peer DOTS agent.

8.2. From DOTS Clients to DOTS Servers

DOTS clients use PUT requests to signal pre-or-ongoing-mitigation telemetry to DOTS servers. An example of such a request is shown in Figure 36.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": [
      {
        "target": {
          "target-prefix": [
            "2001:db8::1/128"
          ]
        },
        "total-attack-traffic-protocol": [
          {
            "protocol": 17,
            "unit": "megabit-ps",
            "mid-percentile-g": "900"
          }
        ],
        "attack-detail": [
          {
            "vendor-id": 32473,
            "attack-id": 77,
            "start-time": "1608336568",
            "attack-severity": "high"
          }
        ]
      }
    ]
  }
}
```

Figure 36: PUT to Send Pre-or-Ongoing-Mitigation Telemetry, depicted as per Section 5.6

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

The following additional Uri-Path parameter is defined:

tmid: Telemetry Identifier is an identifier for the DOTS pre-or-

ongoing-mitigation telemetry data represented as an integer. This identifier MUST be generated by DOTS clients. 'tmid' values MUST increase monotonically whenever a DOTS client needs to convey new set of pre-or-ongoing-mitigation telemetry.

The procedure specified in Section 4.4.1 of [RFC9132] for 'mid' rollover MUST be followed for 'tmid' rollover.

This is a mandatory attribute. 'tmid' MUST appear after 'cuid' in the Uri-Path options.

'cuid' and 'tmid' MUST NOT appear in the PUT request message body.

At least the 'target' attribute and another pre-or-ongoing-mitigation attribute (Section 8.1) MUST be present in the PUT request. If only the 'target' attribute is present, this request is handled as per Section 8.3.

The relative order of two PUT requests carrying DOTS pre-or-ongoing-mitigation telemetry from a DOTS client is determined by comparing their respective 'tmid' values. If two such requests have an overlapping 'target', the PUT request with higher numeric 'tmid' value will override the request with a lower numeric 'tmid' value. The overlapped lower numeric 'tmid' MUST be automatically deleted and no longer be available.

The DOTS server indicates the result of processing a PUT request using CoAP Response Codes. In particular, the 2.04 (Changed) Response Code is returned if the DOTS server has accepted the pre-or-ongoing-mitigation telemetry. The 5.03 (Service Unavailable) Response Code is returned if the DOTS server has erred. 5.03 uses the Max-Age Option to indicate the number of seconds after which to retry.

How long a DOTS server maintains a 'tmid' as active or logs the enclosed telemetry information is implementation specific. Note that if a 'tmid' is still active, then logging details are updated by the DOTS server as a function of the updates received from the peer DOTS client.

A DOTS client that lost the state of its active 'tmid's or has to set 'tmid' back to zero (e.g., crash or restart) MUST send a GET request to the DOTS server to retrieve the list of active 'tmid' values. The DOTS client may then delete 'tmid's that should not be active anymore (Figure 37). Sending a DELETE with no 'tmid' indicates that all 'tmid's must be deactivated (Figure 38).

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
```

Figure 37: Delete a Pre-or-Ongoing-Mitigation Telemetry

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 38: Delete All Pre-or-Ongoing-Mitigation Telemetry

8.3. From DOTS Servers to DOTS Clients

The pre-or-ongoing-mitigation data (attack details, in particular) can also be signaled from DOTS servers to DOTS clients. For example, a DOTS server co-located with a DDoS detector can collect monitoring information from the target network, identify a DDoS attack using statistical analysis or deep learning techniques, and signal 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 operations 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.

In order to receive pre-or-ongoing-mitigation telemetry notifications from a DOTS server, a DOTS client MUST send a PUT (followed by a GET) with the target filter. An example of such a PUT request is shown in Figure 39. In order to avoid maintaining a long list of such requests, it is RECOMMENDED that DOTS clients include all targets in the same request (assuming this fits within one single datagram). DOTS servers may be instructed to restrict the number of pre-or-ongoing-mitigation requests per DOTS client domain. The pre-or-ongoing mitigation requests MUST be maintained in an active state by the DOTS server until a delete request is received from the same DOTS client to clear this pre-or-ongoing-mitigation telemetry or when the DOTS client is considered inactive (e.g., Section 3.5 of [RFC8783]).

The relative order of two PUT requests carrying DOTS pre-or-ongoing-mitigation telemetry from a DOTS client is determined by comparing their respective 'tmid' values. If such two requests have

overlapping 'target', the PUT request with higher numeric 'tmid' value will override the request with a lower numeric 'tmid' value. The overlapped lower numeric 'tmid' MUST be automatically deleted and no longer be available.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=567"
Content-Format: "application/dots+cbor"
```

```
{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": [
      {
        "target": {
          "target-prefix": [
            "2001:db8::/32"
          ]
        }
      }
    ]
  }
}
```

Figure 39: PUT to Request Pre-or-Ongoing-Mitigation Telemetry, depicted as per Section 5.6

DOTS clients of the same domain can request to receive pre-or-ongoing-mitigation telemetry bound to the same target without being considered to be "overlapping" and in conflict.

Once the PUT request to instantiate request state on the server has succeeded, the DOTS client issues a GET request to receive ongoing telemetry updates. The client uses the Observe Option, set to '0' (register), in the GET request to receive asynchronous notifications carrying pre-or-ongoing-mitigation telemetry data from the DOTS server. The GET request can specify a specific 'tmid' (Figure 40) or omit the 'tmid' (Figure 41) to receive updates on all active requests from that client.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=567"
Observe: 0
```

Figure 40: GET to Subscribe to Telemetry Asynchronous
Notifications for a Specific 'tmid'

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Observe: 0
```

Figure 41: GET to Subscribe to Telemetry Asynchronous
Notifications for All 'tmids'

The DOTS client can use a filter to request a subset of the asynchronous notifications from the DOTS server by indicating one or more Uri-Query options in its GET request. A Uri-Query option can include the following parameters to restrict the notifications based on the attack target: 'target-prefix', 'target-port', 'target-protocol', 'target-fqdn', 'target-uri', 'alias-name', 'mid', and 'c' (content) (Section 5.4). Furthermore:

If more than one Uri-Query option is included in a request, these options are interpreted in the same way as when multiple target attributes are included in a message body (Section 4.4.1 of [RFC9132]).

If multiple values of a query parameter are to be included in a request, these values MUST be included in the same Uri-Query option and separated by a "," character without any spaces.

Range values (i.e., a contiguous inclusive block) can be included for the 'target-port', 'target-protocol', and 'mid' parameters by indicating the two boundary values separated by a "-" character.

Wildcard names (i.e., a name with the leftmost label is the "*" character) can be included in 'target-fqdn' or 'target-uri' parameters. DOTS clients MUST NOT include a name in which the "*" character is included in a label other than the leftmost label. "*.example.com" is an example of a valid wildcard name that can be included as a value of the 'target-fqdn' parameter in an Uri-Query option.

DOTS clients may also filter out the asynchronous notifications from the DOTS server by indicating information about a specific attack source. To that aim, a DOTS client may include 'source-prefix', 'source-port', or 'source-icmp-type' in a Uri-Query option. The same considerations (ranges, multiple values) specified for target attributes apply for source attributes. Special care SHOULD be taken when using these filters as their use may cause some attacks may be hidden to the requesting DOTS client (e.g., if the attack changes its source information).

Requests with invalid query types (e.g., not supported, malformed) received by the DOTS server MUST be rejected with a 4.00 (Bad Request) response code.

An example of a request to subscribe to asynchronous telemetry notifications regarding UDP traffic is shown in Figure 42. This filter will be applied for all 'tmid's.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Query: "target-protocol=17"
Observe: 0
```

Figure 42: GET Request to Receive Telemetry Asynchronous Notifications Filtered using Uri-Query

The DOTS server will send asynchronous notifications to the DOTS client when an attack event is detected following similar considerations as in Section 4.4.2.1 of [RFC9132]. An example of a pre-or-ongoing-mitigation telemetry notification is shown in Figure 43.

```

{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": [
      {
        "tmid": 567,
        "target": {
          "target-prefix": [
            "2001:db8::1/128"
          ]
        },
        "target-protocol": [
          17
        ],
        "total-attack-traffic": [
          {
            "unit": "megabit-ps",
            "mid-percentile-g": "900"
          }
        ],
        "attack-detail": [
          {
            "vendor-id": 32473,
            "attack-id": 77,
            "start-time": "1618339785",
            "attack-severity": "high"
          }
        ]
      }
    ]
  }
}

```

Figure 43: Message Body of a Pre-or-Ongoing-Mitigation Telemetry Notification from the DOTS Server, depicted as per Section 5.6

A DOTS server sends the aggregate data for a target using 'total-attack-traffic' attribute. The aggregate assumes that Uri-Query filters are applied on the target. The DOTS server MAY include more fine-grained data when needed (that is, 'total-attack-traffic-protocol' and 'total-attack-traffic-port'). If a port filter (or protocol filter) is included in a request, 'total-attack-traffic-protocol' (or 'total-attack-traffic-port') conveys the data with the port (or protocol) filter applied.

A DOTS server may aggregate pre-or-ongoing-mitigation data (e.g., 'top-talker') for all targets of a domain, or when justified, send specific information (e.g., 'top-talker') per individual targets.

The DOTS client may log pre-or-ongoing-mitigation telemetry data with an alert sent to an administrator or a network controller. The DOTS client may send a mitigation request if the attack cannot be handled locally.

A DOTS client that is not interested to receive pre-or-ongoing-mitigation telemetry data for a target sends a delete request similar to the one depicted in Figure 37.

9. DOTS Telemetry Mitigation Status Update

9.1. DOTS Clients to Servers Mitigation Efficacy DOTS Telemetry Attributes

The mitigation efficacy telemetry attributes can be signaled from DOTS clients to DOTS servers as part of the periodic mitigation efficacy updates to the server (Section 4.4.3 of [RFC9132]).

Total Attack Traffic: The overall attack traffic as observed from the DOTS client perspective during an active mitigation. See Figure 27.

Attack Details: The overall attack details as observed from the DOTS client perspective during an active mitigation. See Section 8.1.5.

The "ietf-dots-telemetry" YANG module (Section 11.1) augments the 'mitigation-scope' message type defined in the "ietf-dots-signal" module [RFC9132] so that these attributes can be signalled by a DOTS client in a mitigation efficacy update (Figure 44).

```
augment-structure /dots-signal:dots-signal/dots-signal:message-type
                  /dots-signal:mitigation-scope/dots-signal:scope:
+-- total-attack-traffic* [unit]
|   +-- unit                unit
|   +-- low-percentile-g?   yang:gauge64
|   +-- mid-percentile-g?   yang:gauge64
|   +-- high-percentile-g?  yang:gauge64
|   +-- peak-g?             yang:gauge64
|   +-- current-g?          yang:gauge64
+-- attack-detail* [vendor-id attack-id]
|   +-- vendor-id           uint32
|   +-- attack-id           uint32
|   +-- attack-description? string
|   +-- attack-severity?    attack-severity
|   +-- start-time?         uint64
|   +-- end-time?           uint64
```

```

+-- source-count
|   +-- low-percentile-g?    yang:gauge64
|   +-- mid-percentile-g?    yang:gauge64
|   +-- high-percentile-g?   yang:gauge64
|   +-- peak-g?              yang:gauge64
|   +-- current-g?           yang:gauge64
+-- top-talker
|   +-- talker* [source-prefix]
|   |   +-- spoofed-status?    boolean
|   |   +-- source-prefix      inet:ip-prefix
|   |   +-- source-port-range* [lower-port]
|   |   |   +-- lower-port    inet:port-number
|   |   |   +-- upper-port?   inet:port-number
|   |   +-- source-icmp-type-range* [lower-type]
|   |   |   +-- lower-type    uint8
|   |   |   +-- upper-type?   uint8
|   |   +-- total-attack-traffic* [unit]
|   |   |   +-- unit          unit
|   |   |   +-- low-percentile-g?    yang:gauge64
|   |   |   +-- mid-percentile-g?    yang:gauge64
|   |   |   +-- high-percentile-g?   yang:gauge64
|   |   |   +-- peak-g?              yang:gauge64
|   |   |   +-- current-g?           yang:gauge64
|   |   +-- total-attack-connection
|   |   |   +-- connection-c
|   |   |   |   +-- low-percentile-g?    yang:gauge64
|   |   |   |   +-- mid-percentile-g?    yang:gauge64
|   |   |   |   +-- high-percentile-g?   yang:gauge64
|   |   |   |   +-- peak-g?              yang:gauge64
|   |   |   |   +-- current-g?           yang:gauge64
|   |   |   +-- embryonic-c
|   |   |   |   ...
|   |   |   +-- connection-ps-c
|   |   |   |   ...
|   |   |   +-- request-ps-c
|   |   |   |   ...
|   |   |   +-- partial-request-c
|   |   |   |   ...

```

Figure 44: Telemetry Efficacy Update Tree Structure

In order to signal telemetry data in a mitigation efficacy update, it is RECOMMENDED that the DOTS client has already established a DOTS telemetry setup session with the server in 'idle' time. Such a session is primarily meant to assess whether the peer DOTS server supports telemetry extensions and, thus, prevent message processing failure (Section 3.1 of [RFC9132]).

An example of an efficacy update with telemetry attributes is depicted in Figure 45.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=123"
If-Match:
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "alias-name": [
          "https1",
          "https2"
        ],
        "attack-status": "under-attack",
        "ietf-dots-telemetry:total-attack-traffic": [
          {
            "unit": "megabit-ps",
            "mid-percentile-g": "900"
          }
        ]
      }
    ]
  }
}
```

Figure 45: An Example of Mitigation Efficacy Update with Telemetry Attributes, depicted as per Section 5.6

9.2. DOTS Servers to Clients 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 4.4.2 of [RFC9132]). In particular, DOTS clients can receive asynchronous notifications of the attack details from DOTS servers using the Observe option defined in [RFC7641].

In order to make use of this feature, DOTS clients MUST establish a telemetry session with the DOTS server in 'idle' time and MUST set the 'server-originated-telemetry' attribute to 'true'.

DOTS servers MUST NOT include telemetry attributes in mitigation status updates sent to DOTS clients for telemetry sessions in which the 'server-originated-telemetry' attribute is set to 'false'.

As defined in [RFC8612], the actual mitigation activities can include several countermeasure mechanisms. The DOTS server signals the current operational status of relevant countermeasures. A list of attacks detected by these countermeasures MAY also be included. The same attributes defined in Section 8.1.5 are applicable for describing the attacks detected and mitigated at the DOTS server domain.

The "ietf-dots-telemetry" YANG module (Section 11.1) augments the 'mitigation-scope' message type defined in "ietf-dots-signal" [RFC9132] with telemetry data as depicted in Figure 46.

```

augment-structure /dots-signal:dots-signal/dots-signal:message-type
                  /dots-signal:mitigation-scope/dots-signal:scope:
+-- (direction)?
|
|  +--:(server-to-client-only)
|  |
|  |  +-- total-traffic* [unit]
|  |  |
|  |  |  +-- unit                                unit
|  |  |  +-- low-percentile-g?                    yang:gauge64
|  |  |  +-- mid-percentile-g?                     yang:gauge64
|  |  |  +-- high-percentile-g?                    yang:gauge64
|  |  |  +-- peak-g?                               yang:gauge64
|  |  |  +-- current-g?                           yang:gauge64
|  |  |
|  |  |  +-- total-attack-connection
|  |  |  |
|  |  |  |  +-- connection-c
|  |  |  |  |
|  |  |  |  |  +-- low-percentile-g?                yang:gauge64
|  |  |  |  |  +-- mid-percentile-g?                 yang:gauge64
|  |  |  |  |  +-- high-percentile-g?                yang:gauge64
|  |  |  |  |  +-- peak-g?                           yang:gauge64
|  |  |  |  |  +-- current-g?                         yang:gauge64
|  |  |  |  |
|  |  |  |  |  +-- embryonic-c
|  |  |  |  |  |
|  |  |  |  |  |  ...
|  |  |  |  |  +-- connection-ps-c
|  |  |  |  |  |
|  |  |  |  |  |  ...
|  |  |  |  |  +-- request-ps-c
|  |  |  |  |  |
|  |  |  |  |  |  ...
|  |  |  |  |  +-- partial-request-c
|  |  |  |  |  |
|  |  |  |  |  |  ...
|  |  |
|  |  |  +-- total-attack-traffic* [unit]
|  |  |  |
|  |  |  |  +-- unit                                unit
|  |  |  |  +-- low-percentile-g?                    yang:gauge64
|  |  |  |  +-- mid-percentile-g?                     yang:gauge64
|  |  |  |  +-- high-percentile-g?                    yang:gauge64
|  |  |  |  +-- peak-g?                               yang:gauge64

```

```

|   +--- current-g?          yang:gauge64
+--- attack-detail* [vendor-id attack-id]
|   +--- vendor-id           uint32
|   +--- attack-id           uint32
|   +--- attack-description?  string
|   +--- attack-severity?     attack-severity
|   +--- start-time?          uint64
|   +--- end-time?            uint64
|   +--- source-count
|   |   +--- low-percentile-g? yang:gauge64
|   |   +--- mid-percentile-g? yang:gauge64
|   |   +--- high-percentile-g? yang:gauge64
|   |   +--- peak-g?           yang:gauge64
|   |   +--- current-g?        yang:gauge64
+--- top-talker
|   +--- talker* [source-prefix]
|   |   +--- spoofed-status?    boolean
|   |   +--- source-prefix      inet:ip-prefix
|   |   +--- source-port-range* [lower-port]
|   |   |   +--- lower-port      inet:port-number
|   |   |   +--- upper-port?     inet:port-number
|   |   +--- source-icmp-type-range* [lower-type]
|   |   |   +--- lower-type      uint8
|   |   |   +--- upper-type?     uint8
|   |   +--- total-attack-traffic* [unit]
|   |   |   +--- unit            unit
|   |   |   +--- low-percentile-g? yang:gauge64
|   |   |   +--- mid-percentile-g? yang:gauge64
|   |   |   +--- high-percentile-g? yang:gauge64
|   |   |   +--- peak-g?         yang:gauge64
|   |   |   +--- current-g?      yang:gauge64
|   |   +--- total-attack-connection
|   |   |   +--- connection-c
|   |   |   |   +--- low-percentile-g? yang:gauge64
|   |   |   |   +--- mid-percentile-g? yang:gauge64
|   |   |   |   +--- high-percentile-g? yang:gauge64
|   |   |   |   +--- peak-g?          yang:gauge64
|   |   |   |   +--- current-g?       yang:gauge64
|   |   |   +--- embryonic-c
|   |   |   |   ...
|   |   |   +--- connection-ps-c
|   |   |   |   ...
|   |   |   +--- request-ps-c
|   |   |   |   ...
|   |   |   +--- partial-request-c
|   |   |   |   ...

```

Figure 46: DOTS Servers to Clients Mitigation Status Telemetry Tree Structure

Figure 47 shows an example of an asynchronous notification of attack mitigation status from the DOTS server. This notification signals both the mid-percentile value of processed attack traffic and the peak count of unique sources involved in the attack.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 12332,
        "mitigation-start": "1507818434",
        "alias-name": [
          "https1",
          "https2"
        ],
        "lifetime": 1600,
        "status": "attack-successfully-mitigated",
        "bytes-dropped": "134334555",
        "bps-dropped": "43344",
        "pkts-dropped": "333334444",
        "pps-dropped": "432432",
        "ietf-dots-telemetry:total-attack-traffic": [
          {
            "unit": "megabit-ps",
            "mid-percentile-g": "752"
          }
        ],
        "ietf-dots-telemetry:attack-detail": [
          {
            "vendor-id": 32473,
            "attack-id": 77,
            "source-count": {
              "peak-g": "12683"
            }
          }
        ]
      }
    ]
  }
}
```

Figure 47: Response Body of a Mitigation Status With Telemetry Attributes, depicted as per Section 5.6

DOTS clients can filter out the asynchronous notifications from the DOTS server by indicating one or more Uri-Query options in its GET request. A Uri-Query option can include the following parameters: 'target-prefix', 'target-port', 'target-protocol', 'target-fqdn', 'target-uri', 'alias-name', and 'c' (content) (Section 5.4). The considerations discussed in Section 8.3 MUST be followed to include multiple query values, ranges ('target-port', 'target-protocol'), and wildcard names ('target-fqdn', 'target-uri').

An example of request to subscribe to asynchronous notifications bound to the "https1" alias is shown in Figure 48.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=12332"
Uri-Query: "target-alias=https1"
Observe: 0
```

Figure 48: GET Request to Receive Asynchronous Notifications
Filtered using Uri- Query

If the target query does not match the target of the enclosed 'mid' as maintained by the DOTS server, the latter MUST respond with a 4.04 (Not Found) error Response Code. The DOTS server MUST NOT add a new observe entry if this query overlaps with an existing one. In such a case, the DOTS server replies with 4.09 (Conflict).

10. Error Handling

A list of common CoAP errors that are implemented by DOTS servers are provided in Section 9 of [RFC9132]. The following additional error cases apply for the telemetry extension:

- * 4.00 (Bad Request) is returned by the DOTS server when the DOTS client has sent a request that violates the DOTS telemetry extension.
- * 4.04 (Not Found) is returned by the DOTS server when the DOTS client is requesting a 'tsid' or 'tmid' that is not valid.
- * 4.00 (Bad Request) is returned by the DOTS server when the DOTS client has sent a request with invalid query types (e.g., not supported, malformed).

- * 4.04 (Not Found) is returned by the DOTS server when the DOTS client has sent a request with a target query that does not match the target of the enclosed 'mid' as maintained by the DOTS server.

As indicated in Section 9 of [RFC9132], an additional plain text diagnostic payload (Section 5.5.2 of [RFC7252]) to help troubleshooting is returned in the body of the response.

11. YANG Modules

11.1. DOTS Signal Channel Telemetry YANG Module

This module uses types defined in [RFC6991] and [RFC8345].

```
<CODE BEGINS> file "ietf-dots-telemetry@2022-02-04.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 dots-signal;
    reference
      "RFC 9132: Distributed Denial-of-Service Open Threat Signaling
      (DOTS) Signal Channel Specification";
  }
  import ietf-dots-data-channel {
    prefix data-channel;
    reference
      "RFC 8783: 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";
  }
}
```

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

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.K
            <mailto:kondtir@gmail.com>";
description
  "This module contains YANG definitions for the signaling
  of DOTS telemetry data exchanged between a DOTS client and
  a DOTS server by means of the DOTS signal channel.

  Copyright (c) 2022 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 Revised BSD License set
  forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).
```

```
        description
            "No effect on the DOTS client domain.";
    }
    enum low {
        value 2;
        description
            "Minimal effect on the DOTS client domain.";
    }
    enum medium {
        value 3;
        description
            "A subset of DOTS client domain resources are
             out of service.";
    }
    enum high {
        value 4;
        description
            "The DOTS client domain is under extremely severe
             conditions.";
    }
    enum unknown {
        value 5;
        description
            "The impact of the attack is not known.";
    }
}
description
    "Enumeration for attack severity.";
reference
    "RFC 7970: The Incident Object Description Exchange
     Format Version 2, Section 3.12.2";
}

typedef unit-class {
    type enumeration {
        enum packet-ps {
            value 1;
            description
                "Packets per second (pps).";
        }
        enum bit-ps {
            value 2;
            description
                "Bits per Second (bit/s).";
        }
    }
    enum byte-ps {
        value 3;
        description
```

```
        "Bytes per second (Byte/s).";
    }
}
description
    "Enumeration to indicate which unit class is used.
    These classes are supported: pps, bit/s, and Byte/s.";
}

typedef unit {
    type enumeration {
        enum packet-ps {
            value 1;
            description
                "Packets per second (pps).";
        }
        enum bit-ps {
            value 2;
            description
                "Bits per Second (bps).";
        }
        enum byte-ps {
            value 3;
            description
                "Bytes per second (Bps).";
        }
        enum kilopacket-ps {
            value 4;
            description
                "Kilo packets per second (kpps).";
        }
        enum kilobit-ps {
            value 5;
            description
                "Kilobits per second (kbps).";
        }
        enum kilobyte-ps {
            value 6;
            description
                "Kilobytes per second (kBps).";
        }
        enum megapacket-ps {
            value 7;
            description
                "Mega packets per second (Mpps).";
        }
        enum megabit-ps {
            value 8;
            description
```

```
        "Megabits per second (Mbps).";
    }
    enum megabyte-ps {
        value 9;
        description
            "Megabytes per second (MBps).";
    }
    enum gigapacket-ps {
        value 10;
        description
            "Giga packets per second (Gpps).";
    }
    enum gigabit-ps {
        value 11;
        description
            "Gigabits per second (Gbps).";
    }
    enum gigabyte-ps {
        value 12;
        description
            "Gigabytes per second (GBps).";
    }
    enum terapacket-ps {
        value 13;
        description
            "Tera packets per second (Tpps).";
    }
    enum terabit-ps {
        value 14;
        description
            "Terabits per second (Tbps).";
    }
    enum terabyte-ps {
        value 15;
        description
            "Terabytes per second (TBps).";
    }
    enum petapacket-ps {
        value 16;
        description
            "Peta packets per second (Ppps).";
    }
    enum petabit-ps {
        value 17;
        description
            "Petabits per second (Pbps).";
    }
    enum petabyte-ps {
```

```
        value 18;
        description
            "Petabytes per second (PBps).";
    }
    enum exapacket-ps {
        value 19;
        description
            "Exa packets per second (Epps).";
    }
    enum exabit-ps {
        value 20;
        description
            "Exabits per second (Ebps).";
    }
    enum exabyte-ps {
        value 21;
        description
            "Exabytes per second (EBps).";
    }
    enum zettapacket-ps {
        value 22;
        description
            "Zetta packets per second (Zpps).";
    }
    enum zettabit-ps {
        value 23;
        description
            "Zettabits per second (Zbps).";
    }
    enum zettabyte-ps {
        value 24;
        description
            "Zettabytes per second (ZBps).";
    }
}
description
    "Enumeration to indicate which unit is used.
    Only one unit per unit class is used owing to
    unit auto-scaling.";
}

typedef interval {
    type enumeration {
        enum 5-minutes {
            value 1;
            description
                "5 minutes.";
        }
    }
}
```

```
enum 10-minutes {
    value 2;
    description
        "10 minutes.";
}
enum 30-minutes {
    value 3;
    description
        "30 minutes.";
}
enum hour {
    value 4;
    description
        "Hour.";
}
enum day {
    value 5;
    description
        "Day.";
}
enum week {
    value 6;
    description
        "Week.";
}
enum month {
    value 7;
    description
        "Month.";
}
}
description
    "Enumeration to indicate the overall measurement period.";
}

typedef sample {
    type enumeration {
        enum second {
            value 1;
            description
                "A one-second measurement period.";
        }
        enum 5-seconds {
            value 2;
            description
                "5-second measurement period.";
        }
        enum 30-seconds {
```



```
        value 3;
        description
            "30-second measurement period.";
    }
    enum minute {
        value 4;
        description
            "One-minute measurement period.";
    }
    enum 5-minutes {
        value 5;
        description
            "5-minute measurement period.";
    }
    enum 10-minutes {
        value 6;
        description
            "10-minute measurement period.";
    }
    enum 30-minutes {
        value 7;
        description
            "30-minute measurement period.";
    }
    enum hour {
        value 8;
        description
            "One-hour measurement period.";
    }
}
description
    "Enumeration to indicate the sampling period.";
}

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.";
}

typedef query-type {
    type enumeration {
        enum target-prefix {
            value 1;
            description
```

```
        "Query based on target prefix.";
    }
    enum target-port {
        value 2;
        description
            "Query based on target port number.";
    }
    enum target-protocol {
        value 3;
        description
            "Query based on target protocol.";
    }
    enum target-fqdn {
        value 4;
        description
            "Query based on target FQDN.";
    }
    enum target-uri {
        value 5;
        description
            "Query based on target URI.";
    }
    enum target-alias {
        value 6;
        description
            "Query based on target alias.";
    }
    enum mid {
        value 7;
        description
            "Query based on mitigation identifier (mid).";
    }
    enum source-prefix {
        value 8;
        description
            "Query based on source prefix.";
    }
    enum source-port {
        value 9;
        description
            "Query based on source port number.";
    }
    enum source-icmp-type {
        value 10;
        description
            "Query based on ICMP type";
    }
    enum content {
```

```
        value 11;
        description
            "Query based on 'c' Uri-Query option that is used
            to control the selection of configuration
            and non-configuration data nodes.";
        reference
            "Section 4.4.2 of RFC 9132.";
    }
}
description
    "Enumeration of support for query types that can be used
    in a GET request to filter out data. Requests with
    invalid query types (e.g., not supported, malformed)
    received by the DOTS server are rejected with
    a 4.00 (Bad Request) response code.";
}

grouping telemetry-parameters {
    description
        "A grouping that includes a set of parameters that
        are used to prepare the reported telemetry data.

        The grouping indicates a measurement interval,
        a measurement sample period, and low/mid/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.

            The measurement sample value must be less than the
            measurement interval value.";
    }
    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-percentile,
      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-percentile,
      this means high-percentiles are disabled.";
  }
}

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

```
grouping percentile-peak-and-current {
  description
    "Generic grouping for percentile and peak values.";
  uses percentile-and-peak;
  leaf current-g {
    type yang:gauge64;
    description
      "Current value.";
  }
}

grouping unit-config {
  description
    "Generic grouping for unit configuration.";
  list unit-config {
    key "unit";
    description
      "Controls which unit classes are allowed when sharing
      telemetry data.";
    leaf unit {
      type unit-class;
      description
        "Can be packet-ps, bit-ps, or byte-ps.";
    }
    leaf unit-status {
      type boolean;
      mandatory true;
      description
        "Enable/disable the use of the measurement unit class.";
    }
  }
}

grouping traffic-unit {
  description
    "Grouping of traffic as a function of the measurement unit.";
  leaf unit {
    type unit;
    description
      "The traffic can be measured using unit classes: packet-ps,
      bit-ps, or byte-ps. DOTS agents auto-scale to the
      appropriate units (e.g., megabit-ps, kilobit-ps).";
  }
  uses percentile-and-peak;
}

grouping traffic-unit-all {
  description
```

```
        "Grouping of traffic as a function of the measurement unit,
          including current values.";
    uses traffic-unit;
    leaf current-g {
        type yang:gauge64;
        description
            "Current observed value.";
    }
}

grouping traffic-unit-protocol {
    description
        "Grouping of traffic of a given transport protocol as
         a function of the measurement unit.";
    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 parameter contains 6 for TCP,
             17 for UDP, 33 for DCCP, or 132 for SCTP.";
    }
    uses traffic-unit;
}

grouping traffic-unit-protocol-all {
    description
        "Grouping of traffic of a given transport protocol as
         a function of the measurement unit, including current
         values.";
    uses traffic-unit-protocol;
    leaf current-g {
        type yang:gauge64;
        description
            "Current observed value.";
    }
}

grouping traffic-unit-port {
    description
        "Grouping of traffic bound to a port number as
         a function of the measurement unit.";
    leaf port {
        type inet:port-number;
        description
            "Port number used by a transport protocol.";
    }
}
```

```
    }
    uses traffic-unit;
  }

  grouping traffic-unit-port-all {
    description
      "Grouping of traffic bound to a port number as
       a function of the measurement unit, including
       current values.";
    uses traffic-unit-port;
    leaf current-g {
      type yang:gauge64;
      description
        "Current observed value.";
    }
  }
}

grouping total-connection-capacity {
  description
    "Total connection capacities for various types of
     connections, as well as overall capacity. These data nodes 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.";
  }
  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. Embryonic connections are 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 new connections allowed per second
            to the target server.";
    }
    leaf connection-client-ps {
        type uint64;
        description
            "The maximum number of new 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-max {
        type uint64;
        description
            "The maximum number of outstanding partial requests
            that are allowed to the target server.";
    }
    leaf partial-request-client-max {
        type uint64;
        description
            "The maximum number of outstanding partial requests
            that are allowed to the target server per client.";
    }
}

grouping total-connection-capacity-protocol {
    description
        "Total connections capacity per protocol. These data nodes are
        useful to detect resource consuming DDoS attacks.";
    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 connection-percentile-and-peak {
    description
        "A set of data nodes which represent the attack
        characteristics.";
    container connection-c {
        uses percentile-and-peak;
        description
            "The number of simultaneous attack connections to
            the target server.";
    }
    container embryonic-c {
        uses percentile-and-peak;
        description
            "The number of simultaneous embryonic connections to
            the target server.";
    }
    container connection-ps-c {
        uses percentile-and-peak;
        description
            "The number of attack connections per second to
            the target server.";
    }
    container request-ps-c {
        uses percentile-and-peak;
        description
            "The number of attack requests per second to
            the target server.";
    }
    container partial-request-c {
        uses percentile-and-peak;
        description
            "The number of attack partial requests to
            the target server.";
    }
}

grouping connection-all {
    description
        "Total attack connections including current values.";
    container connection-c {
        uses percentile-peak-and-current;
        description
```

```
        "The number of simultaneous attack connections to
        the target server.";
    }
    container embryonic-c {
        uses percentile-peak-and-current;
        description
            "The number of simultaneous embryonic connections to
            the target server.";
    }
    container connection-ps-c {
        uses percentile-peak-and-current;
        description
            "The number of attack connections per second to
            the target server.";
    }
    container request-ps-c {
        uses percentile-peak-and-current;
        description
            "The number of attack requests per second to
            the target server.";
    }
    container partial-request-c {
        uses percentile-peak-and-current;
        description
            "The number of attack partial requests to
            the target server.";
    }
}

grouping connection-protocol {
    description
        "Total 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-percentile-and-peak;
}

grouping connection-port {
    description
        "Total attack connections per port number.";
    leaf protocol {
        type uint8;
        description
```

```
        "The transport protocol.
        Values are taken from the IANA Protocol Numbers registry:
        <https://www.iana.org/assignments/protocol-numbers/>.";
    }
    leaf port {
        type inet:port-number;
        description
            "Port number.";
    }
    uses connection-percentile-and-peak;
}

grouping connection-protocol-all {
    description
        "Total attack connections per protocol, including current
        values.";
    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-all;
}

grouping connection-protocol-port-all {
    description
        "Total attack connections per port number, including current
        values.";
    leaf protocol {
        type uint8;
        description
            "The transport protocol.
            Values are taken from the IANA Protocol Numbers registry:
            <https://www.iana.org/assignments/protocol-numbers/>.";
    }
    leaf port {
        type inet:port-number;
        description
            "Port number.";
    }
    uses connection-all;
}

grouping attack-detail {
    description
        "Various details that describe the ongoing
```

```

    attacks that need 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 Private Enterprise Number
      as registered with IANA.";
    reference
      "IANA: Private Enterprise Numbers";
  }
  leaf attack-id {
    type uint32;
    description
      "Unique identifier assigned by the vendor for the attack.";
  }
  leaf description-lang {
    type string {
      pattern '((( [A-Za-z]{2,3} (-[A-Za-z]{3}) (-[A-Za-z]{3}) )
        + ' {0,2} ) ? | [A-Za-z]{4} | [A-Za-z]{5,8} ) (-[A-Za-z]{4} ) ? '
        + ' ( - ( [A-Za-z]{2} | [0-9]{3} ) ) ? ( - ( [A-Za-z0-9]{5,8} '
        + ' | ( [0-9] [A-Za-z0-9]{3} ) ) * ( - [0-9A-WY-Za-wy-z] '
        + ' ( - ( [A-Za-z0-9]{2,8} ) ) + ) * ( - [Xx] ( - ( [A-Za-z0-9] '
        + ' {1,8} ) ) + ) ? | [Xx] ( - ( [A-Za-z0-9]{1,8} ) ) + | '
        + ' (( [Ee] [Nn] - [Gg] [Bb] - [Oo] [Ee] [Dd] | [Ii] - '
        + ' [Aa] [Mm] [Ii] | [Ii] - [Bb] [Nn] [Nn] | [Ii] - '
        + ' [Dd] [Ee] [Ff] [Aa] [Uu] [Ll] [Tt] | [Ii] - '
        + ' [Ee] [Nn] [Oo] [Cc] [Hh] [Ii] [Aa] [Nn] '
        + ' | [Ii] - [Hh] [Aa] [Kk] | '
        + ' [Ii] - [Kk] [Ll] [Ii] [Nn] [Gg] [Oo] [Nn] | '
        + ' [Ii] - [Ll] [Uu] [Xx] | [Ii] - [Mm] [Ii] [Nn] [Gg] [Oo] | '
        + ' [Ii] - [Nn] [Aa] [Vv] [Aa] [Jj] [Oo] | [Ii] - [Pp] [Ww] [Nn] | '
        + ' [Ii] - [Tt] [Aa] [Oo] | [Ii] - [Tt] [Aa] [Yy] | '
        + ' [Ii] - [Tt] [Ss] [Uu] | [Ss] [Gg] [Nn] - [Bb] [Ee] - [Ff] [Rr] | '
        + ' [Ss] [Gg] [Nn] - [Bb] [Ee] - [Nn] [Ll] | [Ss] [Gg] [Nn] - '
        + ' [Cc] [Hh] - [Dd] [Ee] ) | ( [Aa] [Rr] [Tt] - '
        + ' [Ll] [Oo] [Jj] [Bb] [Aa] [Nn] | [Cc] [Ee] [Ll] - '
        + ' [Gg] [Aa] [Uu] [Ll] [Ii] [Ss] [Hh] | '
        + ' [Nn] [Oo] - [Bb] [Oo] [Kk] | [Nn] [Oo] - '
        + ' [Nn] [Yy] [Nn] | [Zz] [Hh] - [Gg] [Uu] [Oo] [Yy] [Uu] | '
        + ' [Zz] [Hh] - [Hh] [Aa] [Kk] [Kk] [Aa] | [Zz] [Hh] - '
        + ' [Mm] [Ii] [Nn] | [Zz] [Hh] - [Mm] [Ii] [Nn] - '
        + ' [Nn] [Aa] [Nn] | [Zz] [Hh] - [Xx] [Ii] [Aa] [Nn] [Gg] ) ) ) ' ;
    }
    default "en-US";
    description
      "Indicates the language tag that is used for

```

```
        'attack-description'. ";
    reference
        "RFC 5646: Tags for Identifying Languages, Section 2.1";
}
leaf attack-description {
    type string;
    description
        "Textual representation of attack description. Natural
        Language Processing techniques (e.g., word embedding)
        might provide some utility in mapping the attack
        description to an attack type.";
}
leaf attack-severity {
    type attack-severity;
    description
        "Severity level of an attack. How this level is determined
        is implementation-specific.";
}
leaf start-time {
    type uint64;
    description
        "The time the attack started. Start time is represented in
        seconds relative to 1970-01-01T00:00:00Z.";
}
leaf end-time {
    type uint64;
    description
        "The time the attack ended. End time is represented in
        seconds relative to 1970-01-01T00:00:00Z.";
}
container source-count {
    description
        "Indicates the count of unique sources involved
        in the attack.";
    uses percentile-and-peak;
    leaf current-g {
        type yang:gauge64;
        description
            "Current observed value.";
    }
}
}

grouping talker {
    description
        "Defines generic data related to top-talkers.";
    leaf spoofed-status {
        type boolean;
    }
}
```

```
    description
      "When set to 'true', it indicates whether this address
       is spoofed.";
  }
  leaf source-prefix {
    type inet:ip-prefix;
    description
      "IPv4 or IPv6 prefix identifying the attacker(s).";
  }
  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;
      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;
      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.";
    }
}
list total-attack-traffic {
    key "unit";
    description
        "Total attack traffic issued from this source.";
    uses traffic-unit-all;
}
}

grouping top-talker-aggregate {
    description
        "An aggregate of top attack sources. This aggregate is
        typically used when included in a mitigation request.";
    list talker {
        key "source-prefix";
        description
            "Refers to a top-talker that is identified by an IPv4
            or IPv6 prefix identifying the attacker(s).";
        uses talker;
        container total-attack-connection {
            description
                "Total attack connections issued from this source.";
            uses connection-all;
        }
    }
}

grouping top-talker {
    description
        "Top attack sources with detailed per-protocol
        structure.";
    list talker {
        key "source-prefix";
        description
            "Refers to a top-talker that is identified by an IPv4
            or IPv6 prefix identifying the attacker(s).";
        uses talker;
        list total-attack-connection-protocol {
            key "protocol";
            description
                "Total attack connections issued from this source.";
            uses connection-protocol-all;
        }
    }
}
```

```
grouping baseline {
  description
    "Grouping for the telemetry baseline.";
  uses data-channel:target;
  leaf-list alias-name {
    type string;
    description
      "An alias name that points to an IP resource.
       An IP resource can be a router, a host,
       an IoT object, a server, etc.";
  }
  list total-traffic-normal {
    key "unit";
    description
      "Total traffic normal baselines.";
    uses traffic-unit;
  }
  list total-traffic-normal-per-protocol {
    key "unit protocol";
    description
      "Total traffic normal baselines per protocol.";
    uses traffic-unit-protocol;
  }
  list total-traffic-normal-per-port {
    key "unit port";
    description
      "Total traffic normal baselines per port number.";
    uses traffic-unit-port;
  }
  list total-connection-capacity {
    key "protocol";
    description
      "Total connection capacity.";
    uses total-connection-capacity-protocol;
  }
  list total-connection-capacity-per-port {
    key "protocol port";
    description
      "Total connection capacity per port number.";
    leaf port {
      type inet:port-number;
      description
        "The target port number.";
    }
    uses total-connection-capacity-protocol;
  }
}
```



```
grouping pre-or-ongoing-mitigation {
  description
    "Grouping for the telemetry data.";
  list total-traffic {
    key "unit";
    description
      "Total traffic.";
    uses traffic-unit-all;
  }
  list total-traffic-protocol {
    key "unit protocol";
    description
      "Total traffic per protocol.";
    uses traffic-unit-protocol-all;
  }
  list total-traffic-port {
    key "unit port";
    description
      "Total traffic per port number.";
    uses traffic-unit-port-all;
  }
  list total-attack-traffic {
    key "unit";
    description
      "Total attack traffic.";
    uses traffic-unit-all;
  }
  list total-attack-traffic-protocol {
    key "unit protocol";
    description
      "Total attack traffic per protocol.";
    uses traffic-unit-protocol-all;
  }
  list total-attack-traffic-port {
    key "unit port";
    description
      "Total attack traffic per port number.";
    uses traffic-unit-port-all;
  }
  list total-attack-connection-protocol {
    key "protocol";
    description
      "Total attack connections.";
    uses connection-protocol-all;
  }
  list total-attack-connection-port {
    key "protocol port";
    description
```

```
    "Total attack connections per target port number.";
    uses connection-protocol-port-all;
  }
  list attack-detail {
    key "vendor-id attack-id";
    description
      "Provides a set of attack details.";
    uses attack-detail;
    container top-talker {
      description
        "Lists the top attack sources.";
      uses top-talker;
    }
  }
}

sx:augment-structure "/dots-signal:dots-signal"
  + "/dots-signal:message-type"
  + "/dots-signal:mitigation-scope"
  + "/dots-signal:scope" {
  description
    "Extends mitigation scope with telemetry update data.";
  choice direction {
    description
      "Indicates the communication direction in which the
      data nodes can be included.";
    case server-to-client-only {
      description
        "These data nodes appear only in a mitigation message
        sent from the server to the client.";
      list total-traffic {
        key "unit";
        description
          "Total traffic.";
        uses traffic-unit-all;
      }
      container total-attack-connection {
        description
          "Total attack connections.";
        uses connection-all;
      }
    }
  }
  list total-attack-traffic {
    key "unit";
    description
      "Total attack traffic.";
    uses traffic-unit-all;
  }
}
```

```
    }
    list attack-detail {
      key "vendor-id attack-id";
      description
        "Attack details";
      uses attack-detail;
      container top-talker {
        description
          "Top attack sources.";
        uses top-talker-aggregate;
      }
    }
  }
}
sx:structure dots-telemetry {
  description
    "Main structure for DOTS telemetry messages.";
  choice telemetry-message-type {
    description
      "Can be a telemetry-setup or telemetry data.";
    case telemetry-setup {
      description
        "Indicates the message is about telemetry setup.";
      choice direction {
        description
          "Indicates the communication direction in which the
            data nodes can be included.";
        case server-to-client-only {
          description
            "These data nodes appear only in a telemetry message
              sent from the server to the client.";
          container max-config-values {
            description
              "Maximum acceptable configuration values.";
            uses telemetry-parameters;
            leaf server-originated-telemetry {
              type boolean;
              default "false";
              description
                "Indicates whether the DOTS server can be
                  instructed to send pre-or-ongoing-mitigation
                  telemetry. If set to 'false' or the data node
                  is not present, this is an indication that
                  the server does not support this capability.";
            }
          }
          leaf telemetry-notify-interval {
            type uint16 {
              range "1 .. 3600";
            }
          }
        }
      }
    }
  }
}
```

```
    units "seconds";
    must '. >= ../../min-config-values'
      + '/telemetry-notify-interval' {
      error-message
        "The value must be greater than or equal
        to the telemetry-notify-interval in the
        min-config-values";
    }
    description
      "Minimum number of seconds between successive
      telemetry notifications.";
  }
}
container min-config-values {
  description
    "Minimum acceptable configuration values.";
  uses telemetry-parameters;
  leaf telemetry-notify-interval {
    type uint16 {
      range "1 .. 3600";
    }
    units "seconds";
    description
      "Minimum number of seconds between successive
      telemetry notifications.";
  }
}
container supported-unit-classes {
  description
    "Supported unit classes and default activation
    status.";
  uses unit-config;
}
leaf-list supported-query-type {
  type query-type;
  description
    "Indicates which query types are supported by
    the server. If the server does not announce
    the query types it supports, the client will
    be unable to use any of the potential
    query-type values to reduce the returned data
    content from the server.";
}
}
}
list telemetry {
  description
    "The telemetry data per DOTS client. The keys
```

```
of the list are 'cuid' and 'tsid', but these keys are
not represented here because these keys are conveyed
as mandatory Uri-Paths in requests.  Omitting keys
is compliant with RFC8791.";
choice direction {
  description
    "Indicates the communication direction in which the
    data nodes can be included.";
  case server-to-client-only {
    description
      "These data nodes appear only in a telemetry message
      sent from the server to the client.";
    leaf tsid {
      type uint32;
      description
        "A client-assigned 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
      "Used to set telemetry parameters such as setting
      low, mid, and high percentile values.";
    container current-config {
      description
        "Current telemetry configuration values.";
      uses telemetry-parameters;
      uses unit-config;
      leaf server-originated-telemetry {
        type boolean;
        description
          "Used by a DOTS client to enable/disable whether
          it requests pre-or-ongoing-mitigation telemetry
          from the DOTS server.";
      }
    }
    leaf telemetry-notify-interval {
      type uint16 {
        range "1 .. 3600";
      }
      units "seconds";
      description
        "Minimum number of seconds between successive
        telemetry notifications.";
```

```
    }  
  }  
}  
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 of  
        the DOTS client domain.";  
    }  
    leaf capacity {  
      type uint64;  
      mandatory true;  
      description  
        "Pipe capacity. This attribute is mandatory when  
        total-pipe-capacity is included in a message.";  
    }  
    leaf unit {  
      type unit;  
      description  
        "The traffic can be measured using unit classes:  
        packets per second (pps), bits per second  
        (bit/s), and/or bytes per second (Byte/s).  
  
        For a given unit class, the DOTS agents  
        auto-scales to the appropriate units (e.g.,  
        megabit-ps, kilobit-ps).";  
    }  
  }  
}  
case baseline {  
  description  
    "Traffic baseline information of a DOTS client  
    domain.";  
  list baseline {  
    key "id";  
    description  
      "Traffic baseline information of a DOTS client  
      domain.";  
    leaf id {  
      type uint32;  
      must '. >= 1';  
    }  
  }  
}
```

```
        description
        "An identifier that uniquely identifies a
        baseline entry communicated by a DOTS client.";
    }
    uses baseline;
}
}
}
}
case telemetry {
    description
    "Telemetry information.";
    list pre-or-ongoing-mitigation {
        description
        "Pre-or-ongoing-mitigation telemetry per DOTS client.
        The keys of the list are 'cuid' and 'tmid', but these
        keys are not represented here because these keys are
        conveyed as mandatory Uri-Paths in requests.
        Omitting keys is compliant with RFC8791.";
        choice direction {
            description
            "Indicates the communication direction in which the
            data nodes can be included.";
            case server-to-client-only {
                description
                "These data nodes appear only in a telemetry message
                sent from the server to the client.";
                leaf tmid {
                    type uint32;
                    description
                    "A client-assigned identifier for the DOTS
                    telemetry data.";
                }
            }
        }
    }
    container target {
        description
        "Indicates the target. At least one of the attributes
        'target-prefix', 'target-fqdn', 'target-uri',
        'alias-name', or 'mid-list' must be present in the
        target definition.";
        uses data-channel:target;
        leaf-list alias-name {
            type string;
            description
            "An alias name that points to a resource.";
        }
    }
}
```

```

leaf-list mid-list {
    type uint32;
    description
        "Reference a list of associated mitigation
        requests.";
    reference
        "RFC 9132: Distributed Denial-of-Service Open Threat
        Signaling (DOTS) Signal Channel
        Specification, Section 4.4.1";
}
}
uses pre-or-ongoing-mitigation;
}
}
}
}
}
<CODE ENDS>

```

11.2. Vendor Attack Mapping Details YANG Module

```
<CODE BEGINS> file "ietf-dots-mapping@2022-02-04.yang"
module ietf-dots-mapping {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-mapping";
  prefix dots-mapping;

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

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

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

     Author:      Jon Shallow
                  <mailto:supjps-ietf@jpsshallow.com>";
  description
    "This module contains YANG definitions for the sharing
     DDoS attack mapping details between a DOTS client and
```


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

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

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

```
revision 2022-02-04 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Distributed Denial-of-Service Open Threat
      Signaling (DOTS) Telemetry";
}

feature dots-telemetry {
  description
    "This feature indicates that DOTS telemetry data can be
      shared between DOTS clients and servers.";
}

grouping attack-mapping {
  description
    "A set of information used for sharing vendor attack mapping
      information with a peer.";
  list vendor {
    key "vendor-id";
    description
      "Vendor attack mapping information of the client/server";
    leaf vendor-id {
      type uint32;
      description
        "Vendor ID is a security vendor's Private Enterprise Number
          as registered with IANA.";
      reference
        "IANA: Private Enterprise Numbers";
    }
    leaf vendor-name {
      type string;
      description
```

```

    "The name of the vendor (e.g., company A).";
}
leaf description-lang {
  type string {
    pattern '(([A-Za-z]{2,3}(-[A-Za-z]{3}(-[A-Za-z]{3}))'
      + '{0,2})?|[A-Za-z]{4}|[A-Za-z]{5,8})(-[A-Za-z]{4})?'
      + '(-([A-Za-z]{2}|[0-9]{3}))?(-([A-Za-z0-9]{5,8}'
      + '|([0-9][A-Za-z0-9]{3}))) * (-[0-9A-WY-Za-wy-z]'
      + '(-([A-Za-z0-9]{2,8}))+) * (-[Xx](-([A-Za-z0-9]'
      + '{1,8}))+) ?|[Xx](-([A-Za-z0-9]{1,8}))+|'
      + '([Ee][Nn]-[Gg][Bb]-[Oo][Ee][Dd]|[Ii]-'
      + '[Aa][Mm][Ii]|[Ii]-[Bb][Nn][Nn]|[Ii]-'
      + '[Dd][Ee][Ff][Aa][Uu][Ll][Tt]|[Ii]-'
      + '[Ee][Nn][Oo][Cc][Hh][Ii][Aa][Nn]'
      + '|[Ii]-[Hh][Aa][Kk]|'
      + '[Ii]-[Kk][Ll][Ii][Nn][Gg][Oo][Nn]|'
      + '[Ii]-[Ll][Uu][Xx]|[Ii]-[Mm][Ii][Nn][Gg][Oo]|'
      + '[Ii]-[Nn][Aa][Vv][Aa][Jj][Oo]|[Ii]-[Pp][Ww][Nn]|'
      + '[Ii]-[Tt][Aa][Oo]|[Ii]-[Tt][Aa][Yy]|'
      + '[Ii]-[Tt][Ss][Uu]|[Ss][Gg][Nn]-[Bb][Ee]-[Ff][Rr]|'
      + '[Ss][Gg][Nn]-[Bb][Ee]-[Nn][Ll]|[Ss][Gg][Nn]-'
      + '[Cc][Hh]-[Dd][Ee])|([Aa][Rr][Tt]-'
      + '[Ll][Oo][Jj][Bb][Aa][Nn]|[Cc][Ee][Ll]-'
      + '[Gg][Aa][Uu][Ll][Ii][Ss][Hh]|'
      + '[Nn][Oo]-[Bb][Oo][Kk]|[Nn][Oo]-'
      + '[Nn][Yy][Nn]|[Zz][Hh]-[Gg][Uu][Oo][Yy][Uu]|'
      + '[Zz][Hh]-[Hh][Aa][Kk][Kk][Aa]|[Zz][Hh]-'
      + '[Mm][Ii][Nn]|[Zz][Hh]-[Mm][Ii][Nn]-'
      + '[Nn][Aa][Nn]|[Zz][Hh]-[Xx][Ii][Aa][Nn][Gg]))';
  }
  default "en-US";
  description
    "Indicates the language tag that is used for
    'attack-description'.";
  reference
    "RFC 5646: Tags for Identifying Languages, Section 2.1";
}
leaf last-updated {
  type uint64;
  mandatory true;
  description
    "The time the mapping table was updated. It is represented
    in seconds relative to 1970-01-01T00:00:00Z.";
}
list attack-mapping {
  key "attack-id";
  description
    "Attack mapping details.";
}

```

```
    leaf attack-id {
      type uint32;
      description
        "Unique identifier assigned by the vendor for the
        attack.";
    }
    leaf attack-description {
      type string;
      mandatory true;
      description
        "Textual representation of attack description. Natural
        Language Processing techniques (e.g., word embedding)
        might provide some utility in mapping the attack
        description to an attack type.";
    }
  }
}

augment "/data-channel:dots-data/data-channel:dots-client" {
  if-feature "dots-telemetry";
  description
    "Augments the data channel with a vendor attack
    mapping table of the DOTS client.";
  container vendor-mapping {
    description
      "Used by DOTS clients to share their vendor
      attack mapping information with DOTS servers.";
    uses attack-mapping;
  }
}

augment "/data-channel:dots-data/data-channel:capabilities" {
  if-feature "dots-telemetry";
  description
    "Augments the DOTS server capabilities with a
    parameter to indicate whether they can share
    attack mapping details.";
  leaf vendor-mapping-enabled {
    type boolean;
    config false;
    description
      "Indicates that the DOTS server supports sharing
      attack vendor mapping details with DOTS clients.";
  }
}

augment "/data-channel:dots-data" {
```

```

    if-feature "dots-telemetry";
    description
      "Augments the data channel with a vendor attack
       mapping table of the DOTS server.";
    container vendor-mapping {
      config false;
      description
        "Includes the list of vendor attack mapping details
         that will be shared upon request with DOTS clients.";
      uses attack-mapping;
    }
  }
}
<CODE ENDS>

```

12. 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 Table 3:

- * Note: Implementers must check that the mapping output provided by their YANG-to-CBOR encoding schemes is aligned with the content of Table 2.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
tsid	uint32	TBA1	0 unsigned	Number
telemetry	list	TBA2	4 array	Array
low-percentile	decimal64	TBA3	6 tag 4 [-2, integer]	String
mid-percentile	decimal64	TBA4	6 tag 4 [-2, integer]	String
high-percentile	decimal64	TBA5	6 tag 4 [-2, integer]	String
unit-config	list	TBA6	4 array	Array
unit	enumeration	TBA7	0 unsigned	String
unit-status	boolean	TBA8	7 bits 20 7 bits 21	False True
total-pipe-capacity	list	TBA9	4 array	Array
link-id	string	TBA10	3 text string	String
pre-or-ongoing-mitigation	list	TBA11	4 array	Array
total-traffic-normal	list	TBA12	4 array	Array
low-percentile-g	yang:gauge64	TBA13	0 unsigned	String
mid-percentile-g	yang:gauge64	TBA14	0 unsigned	String

high-percentile-g	yang:gauge64	TBA15	0 unsigned	String
peak-g	yang:gauge64	TBA16	0 unsigned	String
total-attack-traffic	list	TBA17	4 array	Array
total-traffic	list	TBA18	4 array	Array
total-connection- capacity	list	TBA19	4 array	Array
connection	uint64	TBA20	0 unsigned	String
connection-client	uint64	TBA21	0 unsigned	String
embryonic	uint64	TBA22	0 unsigned	String
embryonic-client	uint64	TBA23	0 unsigned	String
connection-ps	uint64	TBA24	0 unsigned	String
connection-client-ps	uint64	TBA25	0 unsigned	String
request-ps	uint64	TBA26	0 unsigned	String
request-client-ps	uint64	TBA27	0 unsigned	String
partial-request-max	uint64	TBA28	0 unsigned	String
partial-request- client-max	uint64	TBA29	0 unsigned	String
total-attack- connection	container	TBA30	5 map	Object
connection-c	container	TBA31	5 map	Object
embryonic-c	container	TBA32	5 map	Object
connection-ps-c	container	TBA33	5 map	Object
request-ps-c	container	TBA34	5 map	Object
attack-detail	list	TBA35	4 array	Array
id	uint32	TBA36	0 unsigned	Number
attack-id	uint32	TBA37	0 unsigned	Number
attack-description	string	TBA38	3 text string	String
attack-severity	enumeration	TBA39	0 unsigned	String
start-time	uint64	TBA40	0 unsigned	String
end-time	uint64	TBA41	0 unsigned	String
source-count	container	TBA42	5 map	Object
top-talker	container	TBA43	5 map	Object
spoofed-status	boolean	TBA44	7 bits 20	False
			7 bits 21	True
partial-request-c	container	TBA45	5 map	Object
total-attack- connection-protocol	list	TBA46	4 array	Array
baseline	list	TBA49	4 array	Array
current-config	container	TBA50	5 map	Object
max-config-values	container	TBA51	5 map	Object
min-config-values	container	TBA52	5 map	Object
supported-unit-classes	container	TBA53	5 map	Object
server-originated- telemetry	boolean	TBA54	7 bits 20	False
			7 bits 21	True
telemetry-notify- interval	uint16	TBA55	0 unsigned	Number
tmid	uint32	TBA56	0 unsigned	Number
measurement-interval	enumeration	TBA57	0 unsigned	String

measurement-sample	enumeration	TBA58	0 unsigned	String
talker	list	TBA59	4 array	Array
source-prefix	inet:	TBA60	3 text string	String
	ip-prefix			
mid-list	leaf-list	TBA61	4 array	Array
	uint32		0 unsigned	Number
source-port-range	list	TBA62	4 array	Array
source-icmp-type-range	list	TBA63	4 array	Array
target	container	TBA64	5 map	Object
capacity	uint64	TBA65	0 unsigned	String
protocol	uint8	TBA66	0 unsigned	Number
total-traffic-normal-per-protocol	list	TBA67	4 array	Array
total-traffic-normal-per-port	list	TBA68	4 array	Array
total-connection-capacity-per-port	list	TBA69	4 array	Array
total-traffic-protocol	list	TBA70	4 array	Array
total-traffic-port	list	TBA71	4 array	Array
total-attack-traffic-protocol	list	TBA72	4 array	Array
total-attack-traffic-port	list	TBA73	4 array	Array
total-attack-connection-port	list	TBA74	4 array	Array
port	inet:			
	port-number	TBA75	0 unsigned	Number
supported-query-type	leaf-list	TBA76	4 array	Array
			0 unsigned	String
vendor-id	uint32	TBA77	0 unsigned	Number
ietf-dots-telemetry:telemetry-setup	container	TBA78	5 map	Object
ietf-dots-telemetry:total-traffic	list	TBA79	4 array	Array
ietf-dots-telemetry:total-attack-traffic	list	TBA80	4 array	Array
ietf-dots-telemetry:total-attack-connection	container	TBA81	5 map	Object
ietf-dots-telemetry:attack-detail	list	TBA82	4 array	Array
ietf-dots-telemetry:telemetry	container	TBA83	5 map	Object
current-g	yang:gauge64	TBA84	0 unsigned	String
description-lang	string	TBA85	3 text string	String
lower-type	uint8	32771	0 unsigned	Number

upper-type	uint8	32772	0 unsigned	Number
-----	-----	-----	-----	-----

Table 3: YANG/JSON Mapping Parameters to CBOR

13. IANA Considerations

13.1. DOTS Signal Channel CBOR Key Values

This specification registers the DOTS telemetry attributes in the IANA "DOTS Signal Channel CBOR Key Values" registry [Key-Map].

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

- * Note to the IANA: CBOR keys are assigned from the "128-255" range. This specification meets the requirements listed in Section 3.1 [RFC9132] for assignments in the "128-255" range.
- * Note to the RFC Editor: Please replace all occurrences of "TBA1-TBA84" with the assigned values.

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document(s)
tsid	TBA1	0	IESG	[RFCXXXX]
telemetry	TBA2	4	IESG	[RFCXXXX]
low-percentile	TBA3	6tag4	IESG	[RFCXXXX]
mid-percentile	TBA4	6tag4	IESG	[RFCXXXX]
high-percentile	TBA5	6tag4	IESG	[RFCXXXX]
unit-config	TBA6	4	IESG	[RFCXXXX]
unit	TBA7	0	IESG	[RFCXXXX]
unit-status	TBA8	7	IESG	[RFCXXXX]
total-pipe-capacity	TBA9	4	IESG	[RFCXXXX]
link-id	TBA10	3	IESG	[RFCXXXX]
pre-or-ongoing-mitigation	TBA11	4	IESG	[RFCXXXX]
total-traffic-normal	TBA12	4	IESG	[RFCXXXX]
low-percentile-g	TBA13	0	IESG	[RFCXXXX]
mid-percentile-g	TBA14	0	IESG	[RFCXXXX]
high-percentile-g	TBA15	0	IESG	[RFCXXXX]
peak-g	TBA16	0	IESG	[RFCXXXX]
total-attack-traffic	TBA17	4	IESG	[RFCXXXX]
total-traffic	TBA18	4	IESG	[RFCXXXX]
total-connection-capacity	TBA19	4	IESG	[RFCXXXX]

connection	TBA20	0	IESG	[RFCXXXX]
connection-client	TBA21	0	IESG	[RFCXXXX]
embryonic	TBA22	0	IESG	[RFCXXXX]
embryonic-client	TBA23	0	IESG	[RFCXXXX]
connection-ps	TBA24	0	IESG	[RFCXXXX]
connection-client-ps	TBA25	0	IESG	[RFCXXXX]
request-ps	TBA26	0	IESG	[RFCXXXX]
request-client-ps	TBA27	0	IESG	[RFCXXXX]
partial-request-max	TBA28	0	IESG	[RFCXXXX]
partial-request-client-max	TBA29	0	IESG	[RFCXXXX]
total-attack-connection	TBA30	5	IESG	[RFCXXXX]
connection-c	TBA31	5	IESG	[RFCXXXX]
embryonic-c	TBA32	5	IESG	[RFCXXXX]
connection-ps-c	TBA33	5	IESG	[RFCXXXX]
request-ps-c	TBA34	5	IESG	[RFCXXXX]
attack-detail	TBA35	4	IESG	[RFCXXXX]
id	TBA36	0	IESG	[RFCXXXX]
attack-id	TBA37	0	IESG	[RFCXXXX]
attack-description	TBA38	3	IESG	[RFCXXXX]
attack-severity	TBA39	0	IESG	[RFCXXXX]
start-time	TBA40	0	IESG	[RFCXXXX]
end-time	TBA41	0	IESG	[RFCXXXX]
source-count	TBA42	5	IESG	[RFCXXXX]
top-talker	TBA43	5	IESG	[RFCXXXX]
spoofed-status	TBA44	7	IESG	[RFCXXXX]
partial-request-c	TBA45	5	IESG	[RFCXXXX]
total-attack-connection-protocol	TBA46	4	IESG	[RFCXXXX]
baseline	TBA49	4	IESG	[RFCXXXX]
current-config	TBA50	5	IESG	[RFCXXXX]
max-config-value	TBA51	5	IESG	[RFCXXXX]
min-config-values	TBA52	5	IESG	[RFCXXXX]
supported-unit-classes	TBA53	5	IESG	[RFCXXXX]
server-originated-telemetry	TBA54	7	IESG	[RFCXXXX]
telemetry-notify-interval	TBA55	0	IESG	[RFCXXXX]
tmid	TBA56	0	IESG	[RFCXXXX]
measurement-interval	TBA57	0	IESG	[RFCXXXX]
measurement-sample	TBA58	0	IESG	[RFCXXXX]
talker	TBA59	4	IESG	[RFCXXXX]
source-prefix	TBA60	3	IESG	[RFCXXXX]
mid-list	TBA61	4	IESG	[RFCXXXX]
source-port-range	TBA62	4	IESG	[RFCXXXX]
source-icmp-type-range	TBA63	4	IESG	[RFCXXXX]

target	TBA64	5	IESG	[RFCXXXX]
capacity	TBA65	0	IESG	[RFCXXXX]
protocol	TBA66	0	IESG	[RFCXXXX]
total-traffic-normal-per-protocol	TBA67	4	IESG	[RFCXXXX]
total-traffic-normal-per-port	TBA68	4	IESG	[RFCXXXX]
total-connection-capacity-per-port	TBA69	4	IESG	[RFCXXXX]
total-traffic-protocol	TBA70	4	IESG	[RFCXXXX]
total-traffic-port	TBA71	4	IESG	[RFCXXXX]
total-attack-traffic-protocol	TBA72	4	IESG	[RFCXXXX]
total-attack-traffic-port	TBA73	4	IESG	[RFCXXXX]
total-attack-connection-port	TBA74	4	IESG	[RFCXXXX]
port	TBA75	0	IESG	[RFCXXXX]
supported-query-type	TBA76	4	IESG	[RFCXXXX]
vendor-id	TBA77	0	IESG	[RFCXXXX]
ietf-dots-telemetry:telemetry-setup	TBA78	5	IESG	[RFCXXXX]
ietf-dots-telemetry:total-traffic	TBA79	4	IESG	[RFCXXXX]
ietf-dots-telemetry:total-attack-traffic	TBA80	4	IESG	[RFCXXXX]
ietf-dots-telemetry:total-attack-connection	TBA81	5	IESG	[RFCXXXX]
ietf-dots-telemetry:attack-detail	TBA82	4	IESG	[RFCXXXX]
ietf-dots-telemetry:telemetry	TBA83	5	IESG	[RFCXXXX]
current-g	TBA84	0	IESG	[RFCXXXX]
description-lang	TBA85	3	IESG	[RFCXXXX]

Table 4: Registered DOTS Signal Channel CBOR Key Values

13.2. DOTS Signal Channel Conflict Cause Codes

This specification requests IANA to assign a new code from the "DOTS Signal Channel Conflict Cause Codes" registry [Cause].

Code	Label	Description	Reference
TBA	overlapping-pipes	Overlapping pipe scope	[RFCXXXX]

Table 5: Registered DOTS Signal Channel Conflict Cause Code

- * Note to the RFC Editor: Please replace all occurrences of "TBA" with the assigned value.

13.3. DOTS Signal Telemetry YANG Module

This document requests IANA to register the following URIs 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.

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

This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [RFC6020] 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

name: ietf-dots-mapping
 namespace: urn:ietf:params:xml:ns:yang:ietf-dots-mapping
 maintained by IANA: N
 prefix: dots-mapping
 reference: RFC XXXX

14. Security Considerations

14.1. DOTS Signal Channel Telemetry

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

The DOTS telemetry information includes DOTS client network topology, DOTS client domain pipe capacity, normal traffic baseline and connections' capacity, and threat and mitigation information. Such information is sensitive; it MUST be protected at rest by the DOTS server domain to prevent data leakage. Note that sharing this sensitive data with a trusted DOTS server does not introduce any new significant considerations other than the need for the aforementioned protection. Such a DOTS server is already trusted to have access to that kind of information by being in the position to observe and mitigate attacks.

DOTS clients are typically considered to be trusted devices by the DOTS client domain. DOTS clients may be co-located on network security services (e.g., firewall devices), and a compromised security service potentially can do a lot more damage to the network than just the DOTS client component. This assumption differs from the often held view that devices are untrusted, often referred to as the "zero-trust model". A compromised DOTS client can send fake DOTS telemetry data to a DOTS server to mislead the DOTS server. This attack can be prevented by monitoring and auditing DOTS clients to detect misbehavior and to deter misuse, and by only authorizing the DOTS client to convey DOTS telemetry information for specific target resources (e.g., an application server is authorized to exchange DOTS telemetry for its IP addresses but a DDoS mitigator can exchange DOTS telemetry for any target resource in the network). As a reminder, this is a variation of dealing with compromised DOTS clients as discussed in Section 11 of [RFC9132].

DOTS servers must be capable of defending themselves against DoS attacks from compromised DOTS clients. The following non-comprehensive list of mitigation techniques can be used by a DOTS server to handle misbehaving DOTS clients:

- * The probing rate (defined in Section 4.5 of [RFC9132]) can be used to limit the average data rate to the DOTS server.

- * Rate-limiting DOTS telemetry, including those with new 'tmid' values, from the same DOTS client defends against DoS attacks that would result in varying the 'tmid' to exhaust DOTS server resources. Likewise, the DOTS server can enforce a quota and time-limit on the number of active pre-or-ongoing-mitigation telemetry data items (identified by 'tmid') from the DOTS client.

Note also that telemetry notification interval may be used to rate-limit the pre-or-ongoing-mitigation telemetry notifications received by a DOTS client domain.

14.2. Vendor Attack Mapping

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

All data nodes defined in the YANG module specified in Section 11.2 which can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations to these data nodes without proper protection can have a negative effect on network operations. Appropriate security measures are recommended to prevent illegitimate users from invoking DOTS data channel primitives as discussed in [RFC8783]. Nevertheless, an attacker who can access a DOTS client is technically capable of undertaking various attacks, such as:

- * Communicating invalid attack mapping details to the server ('/data-channel:dots-data/data-channel:dots-client/dots-telemetry:vendor-mapping'), which will mislead the server when correlating attack details.

Some of the readable data nodes in the YANG module specified in Section 11.2 may be considered sensitive. It is thus important to control read access to these data nodes. These are the data nodes and their sensitivity:

- * '/data-channel:dots-data/data-channel:dots-client/dots-telemetry:vendor-mapping' can be misused to infer the DDoS protection technology deployed in a DOTS client domain.
- * '/data-channel:dots-data/dots-telemetry:vendor-mapping' can be used by a compromised DOTS client to leak the attack detection capabilities of the DOTS server. This is a variation of the compromised DOTS client attacks discussed in Section 14.1.

15. Contributors

The following individuals have contributed to this document:

- * Li Su, CMCC, Email: suli@chinamobile.com
- * Pan Wei, Huawei, Email: william.panwei@huawei.com

16. Acknowledgements

The authors would like to thank Flemming Andreassen, Liang Xia, and Kaname Nishizuka, co-authors of [I-D.doron-dots-telemetry], and everyone who had contributed to that document.

Thanks to Kaname Nishizuka, Wei Pan, Yuuhei Hayashi, and Tom Petch for comments and review.

Special thanks to Jon Shallow and Kaname Nishizuka for their implementation and interoperability work.

Many thanks to Jan Lindblad for the yangdoctors review, Nagendra Nainar for the opsdireview, James Gruessing for the artart review, Michael Scharf for the tsv-art review, Ted Lemon for the int-dir review, and Robert Sparks for the gen-art review.

Thanks to Benjamin Kaduk for the detailed AD review.

Thanks to Roman Danyliw, Eric Vyncke, Francesca Palombini, Warren Kumari, Erik Kline, Lars Eggert, and Robert Wilton for the IESG review.

17. References

17.1. Normative References

- [Private-Enterprise-Numbers]
"Private Enterprise Numbers", 4 May 2020,
<<https://www.iana.org/assignments/enterprise-numbers>>.
- [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>>.

- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/info/rfc5646>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [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>>.
- [RFC7970] Danyliw, R., "The Incident Object Description Exchange Format Version 2", RFC 7970, DOI 10.17487/RFC7970, November 2016, <<https://www.rfc-editor.org/info/rfc7970>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [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>>.

- [RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <<https://www.rfc-editor.org/info/rfc8783>>.
- [RFC8791] Bierman, A., Björklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <<https://www.rfc-editor.org/info/rfc8791>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.
- [RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <<https://www.rfc-editor.org/info/rfc9132>>.

17.2. Informative References

- [Cause] IANA, "DOTS Signal Channel Conflict Cause Codes", <<https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-conflict-cause-codes>>.
- [I-D.doron-dots-telemetry]
Doron, E., Reddy, T., Andreassen, F., (Frank), L. X., and K. Nishizuka, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry Specifications", Work in Progress, Internet-Draft, draft-doron-dots-telemetry-00, 30 October 2016, <<https://www.ietf.org/archive/id/draft-doron-dots-telemetry-00.txt>>.
- [I-D.ietf-core-new-block]
Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", Work in Progress, Internet-Draft, draft-ietf-core-new-block-14, 26 May 2021, <<https://www.ietf.org/archive/id/draft-ietf-core-new-block-14.txt>>.

- [I-D.ietf-dots-multihoming] Boucadair, M., Reddy, K., T., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", Work in Progress, Internet-Draft, draft-ietf-dots-multihoming-11, 10 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-dots-multihoming-11.txt>>.
- [I-D.ietf-dots-robust-blocks] Boucadair, M. and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Configuration Attributes for Robust Block Transmission", Work in Progress, Internet-Draft, draft-ietf-dots-robust-blocks-03, 11 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-dots-robust-blocks-03.txt>>.
- [Key-Map] IANA, "DOTS Signal Channel CBOR Key Values", <<https://www.iana.org/assignments/dots/dots.xhtml#dots-signal-channel-cbor-key-values>>.
- [PYANG] "pyang", November 2020, <<https://github.com/mbj4668/pyang>>.
- [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>>.
- [RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <<https://www.rfc-editor.org/info/rfc4732>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [RFC5612] Eronen, P. and D. Harrington, "Enterprise Number for Documentation Use", RFC 5612, DOI 10.17487/RFC5612, August 2009, <<https://www.rfc-editor.org/info/rfc5612>>.
- [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>>.

- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.
- [RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open Threat Signaling (DOTS) Requirements", RFC 8612, DOI 10.17487/RFC8612, May 2019, <<https://www.rfc-editor.org/info/rfc8612>>.
- [RFC8811] Mortensen, A., Ed., Reddy, K., T., Ed., Andreasen, F., Teague, N., and R. Compton, "DDoS Open Threat Signaling (DOTS) Architecture", RFC 8811, DOI 10.17487/RFC8811, August 2020, <<https://www.rfc-editor.org/info/rfc8811>>.
- [RFC8903] Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use Cases for DDoS Open Threat Signaling", RFC 8903, DOI 10.17487/RFC8903, May 2021, <<https://www.rfc-editor.org/info/rfc8903>>.
- [RFC9133] Nishizuka, K., Boucadair, M., Reddy, K., T., and T. Nagata, "Controlling Filtering Rules Using Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", RFC 9133, DOI 10.17487/RFC9133, September 2021, <<https://www.rfc-editor.org/info/rfc9133>>.

Authors' Addresses

Mohamed Boucadair (editor)
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy.K (editor)
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
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

Jon Shallow
United Kingdom
Email: supjps-ietf@jpshallow.com