

DOTS
Internet-Draft
Intended status: Informational
Expires: January 8, 2020

M. Chen
Li. Su
Jin. Peng
CMCC
July 07, 2019

DOTS client carry ddos attack information in signal channel
draft-chen-dots-attack-informations-00

Abstract

This document describes DDoS attack information which can be obtained by DOTS client when the enterprise suspects it is under DDoS attack, these informations will be send from DOTS client to DOTS server using Signal channel within Mitigation Request.

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 January 8, 2020.

Copyright Notice

Copyright (c) 2019 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
2.1. Key Words	3
2.2. Definition of Terms	3
3. Mitigation Use Case 1	4
3.1. directly discard attack flow	4
3.2. Optimal device selection	5
3.3. Optimum path for disposal	5
3.4. Mitigation request parameter	6
4. Mitigation Use Case 2	6
4.1. classified disposal	6
4.2. Standard of Attack Type Definition	7
5. Mitigation Use Case 3	7
5.1. Mitigation alarm baseline	7
6. Mitigation request optional parameters	8
7. Mitigation response parameters	9
8. Security Considerations	10
9. IANA Considerations	10
10. Acknowledgement	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Authors' Addresses	11

1. Introduction

Distributed Denial of Service (DDoS) is a type of resource-consuming attack, which exploits a large number of attack resources and uses standard protocols to attack target objects. DDoS attacks consume a large amount of target network resources or server resources (including computing power, storage capacity, etc.). At present, DDoS attack is one of the most powerful and indefensible attacks on the Internet, and due to the extensive use of mobile devices and IoT devices in recent years, it is easier for DDoS attackers to attack with real attack sources (broilers).

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. In the architecture draft, it says in the draft the enterprise has a DOTS client, which obtains information about the DDoS attack, and signals the DOTS server for help in mitigating the attack. but it doesn't says what the information of DDoS attack is. the scope of this draft is about the information of DDoS attack which DOTS client can obtain.

In the architecture draft, it says in the draft the client signal may also include telemetry information about the attack, if the DOTS client has such information available. But in the signal channel draft it doesn't define optional parameter about the telemetry information which will be regarded as DDoS portrait information.

"DDoS portrait information" is defined as the collection of attributes characterizing the actual attacks that have been detected and mitigated. The DDoS portrait information is an optional set of attributes that can be signaled in the DOTS signal channel. The portrait can be optionally sent from the DOTS Client to Server and vice versa.

This document will divide two directions, before mitigation request and after mitigation is complete. Before mitigation request, DOTS client can obtain informations of attack; After mitigation, DOTS server can obtain from mitigator.

2. Terminology

2.1. Key Words

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]

2.2. Definition of Terms

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:

Attack-bandwidth: the amount of traffic under attack, it is usually expressed numerically.

Flow clean: one selection of Attack traffic deposition, the operation contains recognize, discard and reinage.

Attack Type: used to distinguish between different methods of ddos attack.

Attack type definition: General definition method, Covers most current attack types.

3. Mitigation Use Case 1

3.1. directly discard attack flow

when attack target is under attack, it has to make corresponding disposal, there are two options for disposal, one is blackhole directly which may be take effect in routers, in this way all the attack flow will be discarded by router upper path of attack target, this means that the attack target will not receive any traffic during the attack, all the traffic forwards attack target will be discarded, this has a huge impact on the work environment, especially the host that provide external service. The other way of the disposition is to drainage all the traffic flow to clean center from router, then the clean center will use pattern matching or any other method to find out the attack traffic flow to discard, finally, clean center reinage the normal business traffic back to attack target by upper router, the whole process above is defined as flow clean(Figure 1).

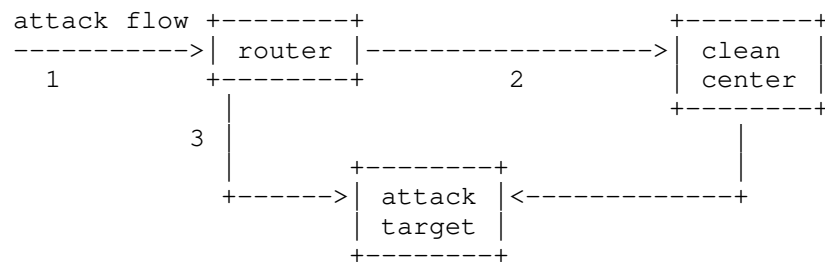


Figure 1: diagram of DDoS Mitigation usecase

Generally, the bandwidth of the link 1 must be larger than link 2 and link 3, and the clean ability of clean center limited to hardware resources. An example of link situation is as below(Figure 2):

figure tag	bandwidth/capability
link 1	100Gb
link 2	50Gb
link 3	10Gb
clean center	80Gb

Figure 2: an example of link bandwidth

The Figure2 is a scenario of the link bandwidth, when a ddos attack is ongoing, if the link 1 bandwidth is completely jammed, the best way to mitigate the attack is to discard all the attack flow; if the amount of the traffic flow is lower than the remainder cleaning ability, the most suitable deposition is to drainage all the attack flow to clean center. Therefore, it is an obvious requirement in the current network environment.

3.2. Optimal device selection

Mitigator may owns a cleaning device cluster and can manage cleaning devices. The capacity of each cleaning equipment is not the same, usually each cleaning equipment utilization rate is not the same, then the remaining cleaning capacity is not consistent. When the attack flow is less than the ability of a cleaning equipment, according to the attack-bandwidth can choose a suitable cleaning equipment, that is conducive to the utilization of equipment; When the attack flow is larger than the cleaning capacity of one cleaning device, several cleaning devices can be optimally scheduled according to the attack-bandwidth.

3.3. Optimum path for disposal

When mitigator is an attack flow cleaning service, they typically deployed the mitigator in a distributed way because of the cost of bandwidth usage with their own leased operator's link bandwidth, and choosing the best traction path was the key to profitability. If the parameter of attack-bandwidth is carried, then the generation of the best drainage path is very meaningful.

When mitigator is at the upstream service operator level, they might have multiple networks, with the attack alert using one network and the flow drainage using another, and the link load is not the same, then carrying the attack-bandwidth is very beneficial for choosing the drainage path, mainly for link load balancing.

3.4. Mitigation request parameter

When a DOTS client requires mitigation for some reason, the DOTS client uses the CoAP PUT method to send a mitigation request to its DOTS server(s). If a DOTS client is entitled to solicit the DOTS service, the DOTS server enables mitigation on behalf of the DOTS client by communicating the DOTS client's request to a mitigator (which may be colocated with the DOTS server) and relaying the feedback of the thus-selected mitigator to the requesting DOTS client.

DOTS clients use the PUT method to request mitigation from a DOTS server. During active mitigation, DOTS clients may use PUT requests to carry mitigation efficacy updates to the DOTS server. We suggest to add attack bandwidth to satisfy the requirement.

total traffic when ddos attack occur, The recommended format is numerical form, such as xxGb. Different attack has different attack bandwidth, numerical value directly reflects the urgency of the current attack. Serious attacks are treated with blackhole, Other cases use flow cleaning, attack-bandwidth is conducive to the selection of disposal mode.

This is an optional attribute.

4. Mitigation Use Case 2

4.1. classified disposal

DDoS attack is a hybrid attack across multiple protocol layers and multiple method, when we deal with DDoS attacks, we find it more reasonable and effective to deal with them according to the types of attacks, It is easier to handle if the type of attack is already included in the mitigation request. There is no doubt that the information may not be accurate, but we can take it as a reference. Therefore, with attack type the disposal process is more helpful. From the point of view of cleaning, different types of attacks are handled differently, for example, Memcached reflection flood use UDP 11211 port for DDoS flood, but tcp syn flood use defects of TCP three-way handshake to consuming connection resources. This two attacks are cleaned in different ways. We suggest to add attack type to satisfy the requirement.

A list of attack types involved in an attack.

There is no uniform definition of attack types, It is often the case that the same type of attack has different names, An attack type is defined in section 4.

The parameter of Target-attack-type contains two value, one is Attack-Name, the other is Attack-Alias, Attack-Alias will solve the abbreviation problem. An attack could be a hybrid attack, then the target-attack-type represents major types of attacks

This is an optional attribute.

4.2. Standard of Attack Type Definition

For the target-attack-type field, we define it as a string Type, and define the two fields according to the attack method and extension name. there may be problems in the actual network environment, that attack target and mitigator (such as cleaning equipment) belong to different models of different vendors, because different vendors have different definitions of Attack in understanding and implementation. When an attack occurs, some devices may not be considered as an attack. It is also possible that the detection device considers it as A type attack, while the cleaning device considers it as B type attack. When performing the cleaning schedule, it will cause the problem of incorrect cleaning or over-cleaning. Both of these errors will cause the normal business to fail to link. Therefore, it is necessary to unify the attack definition, form a standard attack definition, and solve the problem of cleaning errors from the source. we give out a complete format for DDoS attacks as below:

```
[protocol layer] [protocol name] [message name/operation name/port]
[attack methods feature description field 1] [attack methods feature
description field 2] [attack methods describe the standard field]
```

interval between each field operators use special symbol or any other symbol agreed. For example: HTTP Get Flood(CC) definition, we defined the target-Attack-Type field as below (Figure 3):

```
{
  "Attack-Name": " Application_Layer, HTTP, Get,,, Flood"
  "Attack-Alias": "HTTP CC Flood"
}
```

Figure 3: Attack type definition example

5. Mitigation Use Case 3

5.1. Mitigation alarm baseline

Attack target look like to be attacked by DDoS, then DOTS client send mitigation request to DOTS server, So there are exist false alarms. In practice, there are standards for alerting whether or not they are

appropriate, such as alarm baseline. With this parameter, it is possible to determine whether the standard is reasonable or not, False alarms can be corrected and normal alarms can be optimized. It is suggested to use `target_attack_Type_threshold` to carry this information.

DDoS attacks are distributed attacks, it means there are many sources of attack that the traffic from each attack source varies little, so it is more efficient to record the numbers of source ip than the details ip address. Blocking every IP address is a thankless task and short-lived. After mitigation, mitigators can feedback the source ip number to DOTS server, and this information must be more closer to the attack scene.

`target_attack_Type_threshold`: Baseline for a type of attack .

If attack target have the ability to classify each type of DDoS attack, it must have ability to feedback criteria for each type of attack. It doesn't matter that if it can not provide this information, it is just an optional attribute.

This is an optional attribute.

`attack_src_ip_number`: Estimated number of attack sources .

This is an optional attribute.

6. Mitigation request optional parameters

Added parameter show in put method are show as below(Figure 4)

```
Content-Format: "application/dots+cbor"
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "string"
        ],
        "target-port-range": [
          {
            "lower-port": number,
            "upper-port": number
          }
        ],
        "target-protocol": [
          number
        ],
      ]
    ]
  }
}
```



```
    "target-fqdn": [
      "string"
    ],
    "target-Attack-Type": [
      {
        "Attack-Name": ["string"],
        "Attack-Alias": ["string"]
        "target_attack_Type_threshold":["string"]
      }
    ],
    "target-bandwidth":[
      "string"
    ],
    attack_src_ip_number:[
      "string"
    ],

    "target-uri": [
      "string"
    ],
    "alias-name": [
      "string"
    ],
    "lifetime": number,
    "trigger-mitigation": true|false
  }
]
}
```

Figure 4: Mitigation Response for Get Request

7. Mitigation response parameters

After the mitigation of a DDoS attack, DOTS server can obtain some informations from mitigator, these informations are optional parameters only as a suggestion when use DOTS to inform the message between attack target and mitigator. Figure 5 shows a response example of a mitigation request.

```
Content-Format: "application/dots+cbor"
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 12332,
        "mitigation-start": "1507818434",
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-protocol": [
          17
        ],
        "lifetime": 1756,
        "status": "attack-successfully-mitigated",
        "bytes-dropped": "134334555",
        "bps-dropped": "43344",
        "pkts-dropped": "333334444",
        "pps-dropped": "432432",
        "attack_src_ip_number": "5231"
      },
    ]
  }
}
```

Figure 5: PUT to Convey DOTS Mitigation Requests

8. Security Considerations

TBD

9. IANA Considerations

TBD

10. Acknowledgement

TBD

11. References

11.1. 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>>.

11.2. Informative References

- [I-D.ietf-dots-architecture]
Mortensen, A., K, R., Andreasen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-14 (work in progress), May 2019.
- [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-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.
- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.

Authors' Addresses

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

Email: chenmeiling@chinamobile.com

Li Su
CMCC
32, Xuanwumen West
BeiJing 100053
China

Email: suli@chinamobile.com

Jin Peng
CMCC
32, Xuanwumen West
BeiJing 100053
China

Email: pengjin@chinamobile.com

DOTS
Internet-Draft
Intended status: Informational
Expires: January 7, 2020

M. Chen
Li. Su
Jin. Peng
CMCC
July 06, 2019

A method for dots server deployment
draft-chen-dots-server-hierarchical-deployment-00

Abstract

As DOTS is used for DDoS Mitigation signaling, In practice, there are different deployment scenarios for DOTS agents deployment depending on the network deployment mode. This document made an accomodation for DOTS Server deployment which may be Suitable for ISP. 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 January 7, 2020.

Copyright Notice

Copyright (c) 2019 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
5. DOTS server deployment between ISPs	5
6. Security Considerations	6
7. IANA Considerations	6
8. Acknowledgement	6
9. References	6
9.1. Normative References	6
9.2. Informative References	6
Authors' Addresses	7

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], it is says that 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 onfigure DOTS clients with the correct DOTS server(s) nformation 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 id 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.

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.

3. DOTS server Considerations

When take dots server deployment into consideration, one thing must be involved is mitigator. so far, how many network devices can play the role of mitigator, we make a summerized 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

Whether DOTS server can be deployed, the following conditions need to be met:

- o DOTS server has to interconnected with mitigator
- 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 and operations on mitigator
- o DOTS server has the ability to know the address of attack target belong to which mitigator

4. DOTS server deployment inside an ISP

From the internal structure of ISP, the whole network can divide into three parts logically. There are three most important routers: backbone router, man(metropolitan area network) router, and IDC router. When a ddos attack occurs, it must be one of the three cases as follows, and the corresponding mitigator will responsible for mitigation.

- o only the lan network detected the attack, dots server3 will receive mitigation request, and mitigator3 will act as the first responsible mitigator.
- o only the man network detected the attack, dots server2 will receive mitigation request, then mitigator2 will act as the first responsible mitigator.
- o only the backbone network detected the attack, dots server1 will receive mitigation request, then mitigator1 will act as the first responsible mitigator.
- o Attacks on the same attack target are found both in adjacent areas, the upper network mitigator will act as the first responsible mitigator. for example, dots server1 and dots server2 both received the mitigation request from attack target by dots client, mitigator1 will responsible for ddos disposition(priority ranking: mitigator1 > mitigator2 > mitigator3).

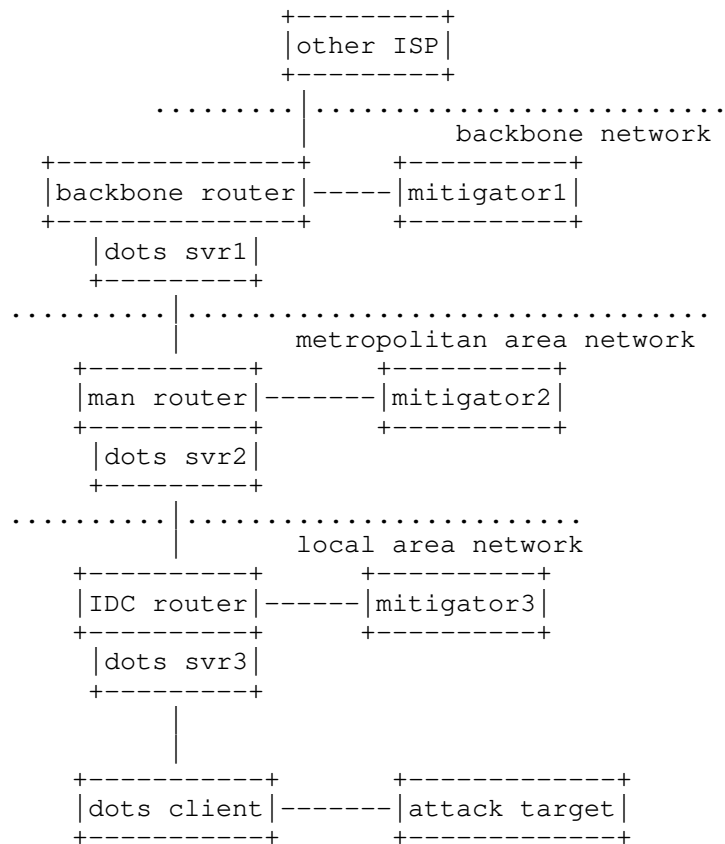


Figure 1: DOTS Server Deployment

5. DOTS server deployment between ISPs

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 C or dots server B will receive the mitigation request.
- o DDOS attack 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.

[I-D.ietf-dots-architecture]

Mortensen, A., K, R., Andreassen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-14 (work in progress), May 2019.

[I-D.ietf-dots-multihoming]

Boucadair, M. and R. K, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-01 (work in progress), January 2019.

[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 R. K, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server Discovery", draft-ietf-dots-server-discovery-04 (work in progress), June 2019.

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

K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.

[I-D.ietf-dots-use-cases]

Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.

Authors' Addresses

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

Email: chenmeiling@chinamobile.com

Li Su
CMCC
32, Xuanwumen West
BeiJing 100053
China

Email: suli@chinamobile.com

Jin Peng
CMCC
32, Xuanwumen West
BeiJing 100053
China

Email: pengjin@chinamobile.com

DOTS
Internet-Draft
Intended status: Informational
Expires: January 6, 2020

Y. Hayashi
NTT
K. Nishizuka
NTT Communications
M. Boucadair
Orange
July 5, 2019

DDoS Mitigation Offload Use Case and DOTS Deployment Considerations
draft-hayashi-dots-dms-offload-usecase-01

Abstract

This document describes a DDoS mitigation offload use case and DOTS deployment consideration of the use case. This use case assumes that a DMS (DDoS Mitigation System) whose utilization rate is high sends its blocked traffic information to an orchestrator using DOTS protocols, then the orchestrator requests forwarding nodes such as routers to filter the traffic. Doing so enables service providers to mitigate DDoS attack traffic automatically while ensuring interoperability and distributed filter enforcement.

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 January 6, 2020.

Copyright Notice

Copyright (c) 2019 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. The Problem	3
4. DDoS Mitigation Offload Use Case	3
5. DOTS Deployment Considerations	5
5.1. DOTS Signaling via Out-of-band Link	7
5.1.1. Example of using Data Channel	7
5.2. DOTS Signaling via In-band Link	8
5.2.1. Example of using Signal Channel	9
5.2.2. Example of using Signal Channel Call Home	11
5.2.3. Data Channel and Signal Channel Controlling Filtering	13
6. Security Considerations	17
7. IANA Considerations	17
8. Acknowledgement	18
9. References	18
9.1. Normative References	18
9.2. Informative References	19
Authors' Addresses	19

1. Introduction

Volume-based distributed denial-of-service (DDoS) attacks such as DNS amplification attacks are critical threats to be handled by service providers. When such attacks occur, service providers have to mitigate them immediately to protect or recover their services.

Therefore, for the service providers to immediately protect their network services from DDoS attacks, DDoS mitigation needs to be automated. To automate DDoS attack mitigation, it is desirable that multi-vendor elements involved in DDoS attack detection and mitigation collaborate 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 between the multi-vendor elements [I-D.ietf-dots-signal-channel] [I-D.ietf-dots-signal-call-home] [I-D.ietf-dots-signal-filter-control] [I-D.ietf-dots-data-channel]. This document describes an automated DDoS Mitigation offload use case

inherited from the DDoS orchestration use case [I-D.ietf-dots-use-cases], which ambitions to enable cost-effective DDoS Mitigation. Furthermore, this document describes deployment consideration for network operators who carry out this use-case using DOTS protocols in their network.

2. Terminology

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

In addition, this document uses the terms defined below:

Mitigation offload: Getting rid of a DMS's mitigation action and assigning the action to another entity when the utilization rate of the DMS reaches a given threshold. How such threshold is set is deployment-specific.

Utilization rate: A scale to measure load of an entity such as link utilization rate or CPU utilization rate.

3. The Problem

In general, DDoS countermeasures are divided into detection and filtering, and detection is technically difficult. DDoS Mitigation System (DMS) can detect attack traffic based on the technology of their vendors, so service providers can increase DDoS countermeasure level by deploying the DMS in their network.

However, the number/capacity of DMS instances that can be deployed in a service providers network is limited due to equipment cost and dimensioning matters. Thus, DMS's utilization rate can reach its maximum capacity faster when the volume of DDoS attacks is enormous. When the rate reaches maximum capacity, the mitigation strategy needs to offload mitigation actions from the DMS to cost-effective forwarding nodes such as routers.

4. DDoS Mitigation Offload Use Case

This section describes offloading mitigation action from DMS whose utilization rate is high to cost-effective forwarding node using DOTS protocols. This section does not consider deployments where the network orchestrator and DMS are co-located.

Figures 1 and 2 show a component diagram and a sequence diagram of the use case, respectively.

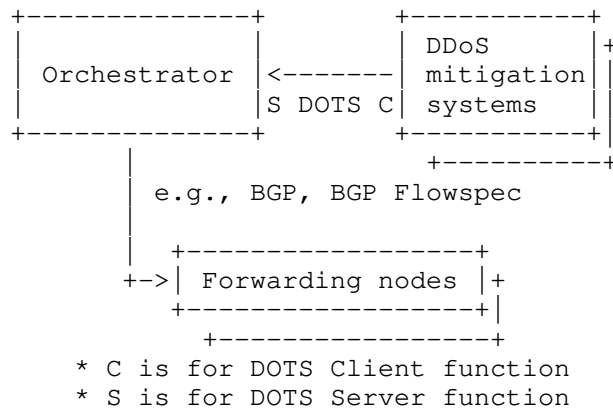


Figure 1: Component Diagram of DDoS Mitigation Offload Use Case

The component diagram shown in Figure 1 differs from that of DDoS Orchestration use case in [I-D.ietf-dots-use-cases] in some respects. First, the DMS embeds a DOTS client to send DOTS requests to the orchestrator. Second, the orchestrator sends a request to underlying forwarding nodes to filter the attack traffic.

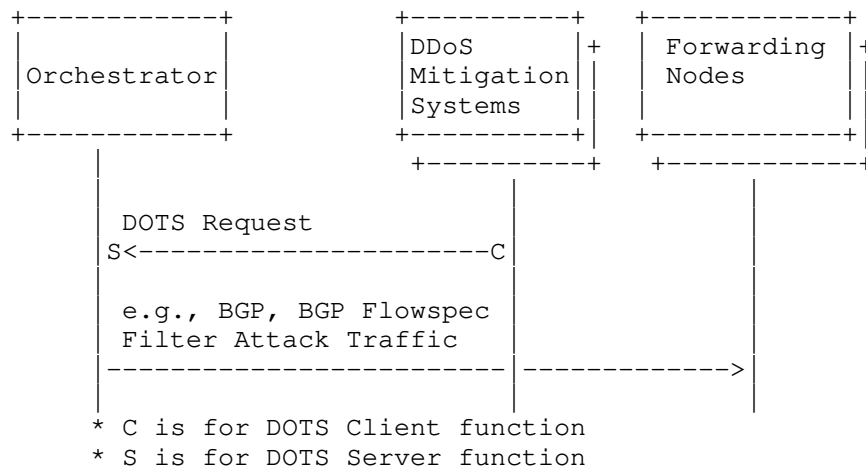


Figure 2: Sequence Diagram of DDoS Mitigation Offload Use Case

In this use case, it is assumed that volume based attack already hits a network and attack traffic is detected and blocked by a DMS in the network. When the volume-based attack becomes intense, DMS's utilization rate can reach a certain threshold (e.g., maximum capacity). Then, the DMS sends a DOTS request as offload request to the orchestrator with the actions to enforce on the traffic. After

that, the orchestrator requests the forwarding nodes to filter attack traffic by dissemination of flow specification rules protocols such as BGP Flowspec [RFC5575] on the basis of the blocked traffic information.

This use case is divided into two cases based on type of link between the DMS and the orchestrator: "out-of-band case" and "in-band case".

"Out-of-band case" is that the DMS sends a DOTS request to the orchestrator with blocked traffic information by the DMS via out-of-band link. The link is not congested when it is under volume attack-time, so the link can convey a lot of information.

On the other hand, "in-band case" is that the DMS sends a mitigation request to the orchestrator with blocked traffic information by the DMS via in-band channel. The link can be congested when it is under volume attack-time, so the link can convey limited information.

5. DOTS Deployment Considerations

This section describes deployment considerations: what type of DOTS protocol can be used and what type of information can be conveyed by DOTS protocol in this use case. Figure 3 shows overview of the DOTS signaling method and conveyed information for the out-of-band case and in-band case.

The volume of information should be considered carefully when DOTS protocol is used in in-band-case. What type of information can be conveyed by DMS relies on attack type detected by the DMS: reflection attack or non-reflection attack. When it is under non-reflection attack, src_ip and src_port information cannot be conveyed because attackers usually randomize the parameters so number of its become enormous. On the other hand, when it is under reflection attack, dst_port information cannot be conveyed because attackers usually randomize src_port so the number of dst_port of attack packets reached to victim become enormous. Furthermore, when it is under reflection attack, src_ip information cannot be conveyed when number of reflector is enormous.

	Reflection Attack	Non-Reflection Attack
Out-of-band case	Attack Time Method : Data Channel Info : src_ip, src_port, dst_ip, dst_port, protocol	
In-band case	Attack Time (Number of reflector is small) Method : Signal Channel Call Home Info : src_ip, src_port, dst_ip, protocol	Attack Time Method : Signal Channel Info : dst_ip, dst_port, protocol
	Attack Time (Number of reflector is enormous) Method : Signal Channel Call Home Info : src_port, dst_ip, protocol	
	Peace Time Method : Data Channel Info : src_port, dst_ip, protocol	Peace Time Method : Data Channel Info : dst_ip, dst_port, protocol
	Attack Time Method : Signal Channel Control Filtering Info : ACL name	Attack Time Method : Signal Channel Control Filtering Info : ACL name

Figure 3: Signaling Method and Conveyed Information

About offloading DMS against reflection attack, the current signal channel [I-D.ietf-dots-signal-channel] is insufficient in terms of conveying src information. On the other hand, both call home expansion [I-D.ietf-dots-signal-call-home] and Filtering control expansion [I-D.ietf-dots-signal-filter-control] can convey src information.

Signal channel expansion of call home defines source-* clauses so it can convey src_ip information and src_port information in attack time. On the other hand, filtering control expansion can activate filtering rule configured in peacetime. Filtering rule for well-known port numbers abused for reflection attack can be configured to

DOTS server in peacetime. However, filtering rule for reflector's ip address in attack time can't be known in peace time. So filtering control expansion can convey src_port information but can't send src_ip information against reflection attack. About sending src information in the DMS offload use case, the capability of the call home extension encompasses the capabilities of the filtering control extension.

Hereafter, this document describes example of use DOTS protocol in each case.

5.1. DOTS Signaling via Out-of-band Link

In this case, the link is not congested when it is under volume attack-time, so DOTS data channel [I-D.ietf-dots-data-channel] is suitable because DOTS data channel has capability of conveying the drop-listed filtering rules including (src_ip, src_port, dst_ip, dst_port, protocol) information (and other actions such as 'rate-limit').

5.1.1. Example of using Data Channel

The procedure to use DOTS Data Channel in such case is as follows:

- o The DMS generates a list of flow (src_ip, src_port, dst_ip, dst_port, protocol) information which the DMS is blocking/rate-limiting and wants to offload.
- o The DMS creates data-channel ACL such as shown figure 4.
- o The DMS sends the data-channel ACL to the orchestrator.

```
{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "DMS_Offload_use_case_ACL",
        "type": "ipv4-acl-type",
        "activation-type": "immediate",
        "aces": {
          "ace": [
            {
              "name": "DMS_Offload_use_case_ACE_00",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "192.0.2.2/32",
                  "source-ipv4-network": "203.0.113.2/32",
                  "protocol": 17
                }
              }
            }
          ]
        }
      }
    ]
  }
}
```


filtering control [I-D.ietf-dots-signal-filter-control] can be used to communicate the policies to the orchestrator.

5.2.1. Example of using Signal Channel

DOTS signal channel has capability to send (dst_ip, dst_port, protocol) information. The procedure to use DOTS Signal Channel in this case is as follows:

- o The DMS generates a list of (dst_ip, dst_port, protocol) information which the DMS is blocking/rate-limiting and wants to offload.
- o The DMS creates mitigation request such as shown figure 5.
- o The DMS sends the mitigation requests to the orchestrator.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          },
          {
            "lower-port": 443
          }
        ],
        "target-protocol": [
          6
        ],
        "lifetime": 3600
      },
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-port-range": [
          {
            "lower-port": 53
          },
          {
            "lower-port": 123
          }
        ],
        "target-protocol": [
          17
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 5: JSON Example of offload request including (dst_ip, dst_port, protocol) information conveyed by DOTS signal channel

5.2.2. Example of using Signal Channel Call Home

DOTS signal channel call home [I-D.ietf-dots-signal-call-home] has capability to send (dst_ip, dst_port, src_ip, src_port, protocol) information. The channel can convey src_ip information when number of reflector detected by DMS is small. The procedure to use DOTS call home in the situation is as follows:

- o The DMS generates a list of (dst_ip, src_ip, src_port, protocol) information which the DMS is blocking/rate-limiting and wants to offload.
- o The DMS creates mitigation request such as shown figure 6.
- o The DMS sends the mitigation requests to the orchestrator.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-protocol": [
          6
        ],
        "source-prefix": [
          "203.0.113.2/32"
        ],
        "source-port-range" : [
          {
            "lower-port": 53
          },
          {
            "lower-port": 123
          }
        ],
        "lifetime": 3600
      },
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-protocol": [
          6
        ],
        "source-prefix": [
          "203.0.113.3/32"
        ]
      }
    ]
  }
}
```

```
    ],
    "source-port-range" : [
      {
        "lower-port": 19
      },
      {
        "lower-port": 11211
      }
    ],
    "lifetime": 3600
  }
]
}
```

Figure 6: JSON Example of offload request including (dst_ip, src_ip, src_port, protocol) information conveyed by DOTS signal channel

On the other hand, signal channel call home cannot convey src_ip information when number of reflector detected by DMS is enormous. The procedure to use DOTS call home in the situation is as follows:

- o The DMS generates a list of (dst_ip, src_port, protocol) information which the DMS is blocking/rate-limiting and wants to offload.
- o The DMS creates mitigation request such as shown figure 7.
- o The DMS sends the mitigation requests to the orchestrator.


```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-protocol": [
          6
        ],
        "source-port-range" : [
          {
            "lower-port": 53
          },
          {
            "lower-port": 123
          },
          {
            "lower-port": 19
          },
          {
            "lower-port": 11211
          }
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 7: JSON Example of offload request including (dst_ip, src_port, protocol) information conveyed by DOTS signal channel

5.2.3. Data Channel and Signal Channel Controlling Filtering

DOTS signal channel controlling filtering

[I-D.ietf-dots-signal-filter-control] has capability to activate or deactivate ACL configured by Data Channel. Against reflection attack, DOTS client configures ACL including (dst_ip, src_port, protocol) information in peace time by Data Channel, and DOTS client activate the ACL in attack time by Signal Channel controlling filtering. Note that the src_port is well known port abused to carry out reflection attack by attacker. The procedure to use DOTS data channel and signal channel controlling filtering is as follows:

- o In peace time, the DMS sends the ACL including (dst_ip, src_port, protocol) information such as figure 8.

- o In attack time, the DMS generates a list of (dst_ip, src_port, protocol) which the DMS is blocking/rate-limiting and wants to offload. After that, the DMS sends the mitigation requests to activate corresponding ACL configured to the orchestrator such as figure 9.

```
{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "DMS_Offload_use_case_ACL",
        "type": "ipv4-acl-type",
        "activation-type": "activate-when-mitigating",
        "aces": {
          "ace": [
            {
              "name": "DMS_Offload_use_case_ACL_DNS_amp",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "192.0.2.2/32",
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 53
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            },
            {
              "name": "DMS_Offload_use_case_ACL_NTP_amp",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "192.0.2.2/32",
                  "protocol": 17
                },
                "udp": {
                  "source-port": {
                    "operator": "eq",
                    "port": 123
                  }
                }
              },
              "actions": {
```

```

        "forwarding": "drop"
      }
    ]
  }
}

```

Figure 8: JSON Example of ACL including (dst_ip, src_port, protocol) information conveyed by DOTS data channel

```

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-protocol": [
          17
        ],
        "acl-list": [
          {
            "acl-name": "DMS_Offload_use_case_ACL_DNS_amp",
            "activation-type": "immediate"
          }
        ]
      }
    ]
  }
}

```

Figure 9: JSON Example of including acl name conveyed by DOTS signal channel

Against non-reflection attack, DOTS client configures ACL including (dst_ip, dst_port, protocol) information in peace time by Data Channel, and DOTS client activate the acl in attack time by Signal Channel. Note that the dst_port is well known port abused to carry out non-reclection attack by attacker. The procedure to use DOTS data channel and signal channel controlling filtering is as follows:

- o In peace time, the DMS sends the ACL including (dst_ip, dst_port, protocol) information such as figure 10.

- o In attack time, the DMS generates a list of (dst_ip, dst_port, protocol) which the DMS is blocking/rate-limiting and wants to offload. After that, the DMS sends the mitigation requests to activate corresponding ACL configured to the orchestrator such as figure 11.

```
{
  "ietf-dots-data-channel:acls": {
    "acl": [
      {
        "name": "DMS_Offload_use_case_ACL",
        "type": "ipv4-acl-type",
        "activation-type": "activate-when-mitigating",
        "aces": {
          "ace": [
            {
              "name": "DMS_Offload_use_case_HTTP_GET_Flooding",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "192.0.2.2/32",
                  "protocol": 6
                },
                "tcp": {
                  "destination-port": {
                    "operator": "eq",
                    "port": 80
                  }
                }
              },
              "actions": {
                "forwarding": "drop"
              }
            },
            {
              "name": "DMS_Offload_use_case_SYN_Flooding_FTP",
              "matches": {
                "ipv4": {
                  "destination-ipv4-network": "192.0.2.2/32",
                  "protocol": 6
                },
                "tcp": {
                  "destination-port": {
                    "operator": "eq",
                    "port": 20
                  }
                }
              },
              "actions": {
```

```

        "forwarding": "drop"
      }
    ]
  }
}

```

Figure 10: JSON Example of ACL including (dst_ip, dst_port, protocol) information conveyed by DOTS data channel

```

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "192.0.2.2/32"
        ],
        "target-protocol": [
          6
        ],
        "acl-list": [
          {
            "acl-name": "DMS_Offload_use_case_HTTP_GET_Flooding",
            "activation-type": "immediate"
          }
        ]
      }
    ]
  }
}

```

Figure 11: JSON Example of including ACL name conveyed by DOTS signal channel

6. Security Considerations

Security considerations discussed in [I-D.ietf-dots-data-channel] and [I-D.ietf-dots-signal-channel] are to be taken into account.

7. IANA Considerations

This document does not require any action from IANA.

8. Acknowledgement

Thanks to Tirumaleswar Reddy, Shunsuke Homma for the comments.
Thanks to Koichi Sakurada for demonstrating proof of concepts of this
.

9. References

9.1. Normative References

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

Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-29 (work in progress), May 2019.

[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-signal-call-home]

K, R., Boucadair, M., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home", draft-ietf-dots-signal-call-home-02 (work in progress), May 2019.

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

K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.

[I-D.ietf-dots-signal-filter-control]

Nishizuka, K., Boucadair, M., K, R., and T. Nagata, "Controlling Filtering Rules Using Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", draft-ietf-dots-signal-filter-control-01 (work in progress), May 2019.

[I-D.ietf-dots-use-cases]

Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.

9.2. Informative References

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

Authors' Addresses

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

Email: yuuei.hayashi@gmail.com

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

Email: kaname@nttv6.jp

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

DOTS
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2019

M. Boucadair
Orange
T. Reddy
McAfee
June 26, 2019

Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server
Discovery
draft-ietf-dots-server-discovery-04

Abstract

It may not be possible for a network to determine the cause for an attack, but instead just realize that some resources seem to be under attack. To fill that gap, Distributed-Denial-of-Service Open Threat Signaling (DOTS) allows a network to inform a DOTS server that it is under a potential attack so that appropriate mitigation actions are undertaken.

This document specifies mechanisms to configure DOTS clients with DOTS servers. The discovery procedure also covers the DOTS Signal Channel Call Home.

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 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Why Multiple Discovery Mechanisms?	4
4. Unified DOTS Discovery Procedure	5
5. DHCP Options for DOTS Agent Discovery	7
5.1. DHCPv6 DOTS Options	8
5.1.1. Format of DOTS Reference Identifier Option	8
5.1.2. Format of DOTS Address Option	8
5.1.3. DHCPv6 Client Behavior	9
5.2. DHCPv4 DOTS Options	10
5.2.1. Format of DOTS Reference Identifier Option	10
5.2.2. Format of DOTS Address Option	11
5.2.3. DHCPv4 Client Behavior	12
6. Discovery using Service Resolution	13
7. DNS Service Discovery	15
8. Security Considerations	16
8.1. DHCP	16
8.2. Service Resolution	16
8.3. DNS Service Discovery	16
9. IANA Considerations	17
9.1. DHCPv6 Option	17
9.2. DHCPv4 Option	17
9.3. Application Service & Application Protocol Tags	17
9.3.1. DOTS Application Service Tag Registration	17
9.3.2. DOTS Call Home Application Service Tag Registration	18
9.3.3. signal.udp Application Protocol Tag Registration	18
9.3.4. signal.tcp Application Protocol Tag Registration	18
9.3.5. data.tcp Application Protocol Tag Registration	18
10. Contributors	18
11. Acknowledgements	19
12. References	19
12.1. Normative References	19
12.2. Informative References	20
Authors' Addresses	21

1. Introduction

DDoS Open Threat Signaling (DOTS) [I-D.ietf-dots-architecture] specifies an architecture, in which 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 inhibits the effectiveness of DDoS attack mitigation, DOTS signal channel protocol [I-D.ietf-dots-signal-channel] is meant to carry requests for DDoS attack mitigation, thereby reducing the impact of an attack and leading to more efficient defensive actions in various deployment scenarios such as those discussed in [I-D.ietf-dots-use-cases]. Moreover, DOTS clients can instruct a DOTS server to install filtering rules by means of DOTS data channel [I-D.ietf-dots-data-channel].

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

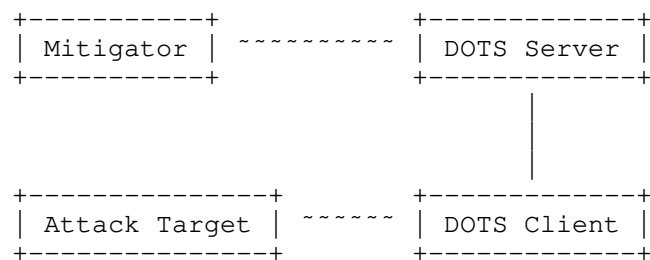


Figure 1: Basic DOTS Architecture

[I-D.ietf-dots-architecture] specifies that the DOTS client may be provided with a list of DOTS servers; each 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 addresses.

This document specifies methods for DOTS clients to discover their DOTS server(s). The rationale for specifying multiple discovery mechanisms is discussed in Section 3.

The discovery methods can also be used by a DOTS server to locate a DOTS client in the context of DOTS Signal Channel Call Home [I-D.ietf-dots-signal-call-home].

Considerations for the selection of DOTS server(s) by multi-homed DOTS clients is out of scope; the reader should refer to [I-D.ietf-dots-multihoming] for more details.

This document assumes that security credentials to authenticate DOTS server(s) are provisioned to a DOTS client using a variety of means such as (but not limited to) those discussed in [I-D.ietf-netconf-zerotouch] or [I-D.ietf-anima-bootstrapping-keyinfra]. DOTS clients use those credentials for authentication purposes following the rules documented in [I-D.ietf-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 [I-D.ietf-dots-architecture] and [RFC3958].

DHCP refers to both DHCPv4 [RFC2131] and DHCPv6 [RFC8415].

"Peer DOTS agent" refers to the peer DOTS server (normal DOTS operation) or to a peer DOTS client (for DOTS Signal Channel Call Home).

3. Why Multiple Discovery Mechanisms?

It is tempting to specify one single discovery mechanism for DOTS. Nevertheless, the analysis of the various use cases sketched in [I-D.ietf-dots-use-cases] reveals that it is unlikely that one single discovery method can be suitable for all the sample deployments. Concretely:

- o Many use cases discussed in [I-D.ietf-dots-use-cases] do involve a CPE device. Multiple CPEs, connected to distinct network providers may even be considered. It is intuitive to leverage on existing mechanisms such as discovery using service resolution or DHCP to provision the CPE acting as a DOTS client with the DOTS server(s).
- o Resolving a DOTS server domain name offered by an upstream transit provider provisioned to a DOTS client into IP address(es) require the use of the appropriate DNS resolvers; otherwise, resolving those names will fail. The use of protocols such as DHCP does

allow to associate provisioned DOTS server domain names with a list of DNS servers to be used for name resolution. Furthermore, DHCP allows to directly provision IP addresses avoiding therefore the need for extra lookup delays.

- o Some of the use cases may allow DOTS clients to have direct communications with upstream DOTS servers; that is no DOTS gateway is involved. Leveraging on existing features that do not require specific feature on the node embedding the DOTS client may ease DOTS deployment. Typically, the use of Straightforward-Naming Authority Pointer (S-NAPTR) lookups [RFC3958] allows the DOTS server administrators to provision the preferred DOTS transport protocol between the DOTS client and the DOTS server and allows the DOTS client to discover this preference.
- o The upstream network provider is not the DDoS mitigation provider for some of these use cases. It is safe to assume that for such deployments, the DOTS server(s) domain name is provided during the service subscription (i.e., manual/local configuration).
- o Multiple DOTS clients may be enabled within a network (e.g., enterprise network). Dynamic means to discover DOTS servers in a deterministic manner are interesting from an operational standpoint.
- o Some of the use cases may involve a DOTS gateway that is responsible for selecting the appropriate DOTS server(s) to relay requests received from DOTS clients.

Consequently, this document describes a unified discovery logic (Section 4) which involves the following mechanisms:

- o Dynamic discovery using DHCP (Section 5).
- o A resolution mechanism based on straightforward Naming Authority Pointer (S-NAPTR) resource records in the Domain Name System (DNS) (Section 6).
- o DNS Service Discovery (Section 7).

4. Unified DOTS Discovery Procedure

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. To accomplish this, operators will need a consistent set of ways in which DOTS clients can discover this information, and a consistent priority among these options. If some devices prefer manual configuration over dynamic discovery,

while others prefer dynamic discovery over manual configuration, the result will be a process of "whack-a-mole", where the operator must find devices that are using the wrong DOTS server(s), determine how to ensure the devices are configured properly, and then reconfigure the device through the preferred method.

All DOTS clients MUST support at least one of the three mechanisms below to determine a DOTS server list. All DOTS clients SHOULD implement all three, or as many as are practical for any specific device, of these ways to discover DOTS servers, in order to facilitate the deployment of DOTS in large scale environments:

1. Explicit configuration:

- * Local/Manual configuration: A DOTS client, will learn the DOTS server(s) by means of local or manual DOTS configuration (i.e., DOTS servers configured at the system level). Configuration discovered from a DOTS client application is considered as local configuration.

An implementation may give the user an opportunity (e.g., by means of configuration file options or menu items) to specify DOTS server(s) for each address family. These MAY be specified either as IP addresses or the DNS name of a DOTS server. When only DOTS server's IP addresses are configured, a reference identifier must also be configured for authentication purposes.

- * Automatic configuration (e.g., DHCP, an automation system): The DOTS client attempts to discover DOTS server(s) names and/or addresses from DHCP, as described in Section 5.

2. Service Resolution : The DOTS client attempts to discover DOTS server name(s) using service resolution, as specified in Section 6.

3. DNS SD: DNS Service Discovery. The DOTS client attempts to discover DOTS server name(s) using DNS service discovery, as specified in Section 7.

Some of these mechanisms imply the use of DNS to resolve the IP address(es) of the DOTS server, while others imply an IP address of the relevant DOTS server is obtained directly. Implementation options may vary on a per device basis, as some devices may not have DNS capabilities and/or proper configuration.

DOTS clients will prefer information received from the discovery methods in the order listed.

On hosts with more than one interface or address family (IPv4/v6), the DOTS server discovery procedure has to be performed for each combination of interface and address family. A client MAY choose to perform the discovery procedure only for a desired interface/address combination if the client does not wish to discover a DOTS server for all combinations of interface and address family.

The above procedure MUST also be followed by a DOTS gateway. Likewise, this procedure MUST be followed by a DOTS server in the context of DOTS Signal Channel Call Home [I-D.ietf-dots-signal-call-home].

The discovery method MUST be reiterated upon the following events:

- o Expiry of a lease associated with a discovered DOTS server.
- o Expiry of a DOTS server's certificate currently in use.
- o Attachment to a new network.

5. DHCP Options for DOTS Agent Discovery

As reported in Section 1.7.2 of [RFC6125]:

"few certification authorities issue server certificates based on IP addresses, but preliminary evidence indicates that such certificates are a very small percentage (less than 1%) of issued certificates".

In order to allow for PKIX-based authentication between a DOTS client and server while accommodating for the current best practices for issuing certificates, this document allows for configuring names to DOTS clients. These names can be used for two purposes: to retrieve the list of IP addresses of a DOTS server or to be presented as a reference identifier for authentication purposes.

Defining the option to include a list of IP addresses would avoid a dependency on an underlying name resolution, but that design requires to also supply a name for PKIX-based authentication purposes.

The design assumes that the same peer DOTS agent is used for establishing both signal and data channels. For more customized configurations (e.g., transport-specific configuration, distinct DOTS servers for the signal and the data channels), an operator can supply only a DOTS reference identifier that will be then passed to the procedure described in Section 6.

5.1. DHCPv6 DOTS Options

5.1.1. Format of DOTS Reference Identifier Option

The DHCPv6 DOTS Reference Identifier option is used to configure a name of the DOTS server (or the name of the DOTS client for DOTS Signal Channel Call Home). The format of this option is shown in Figure 2.

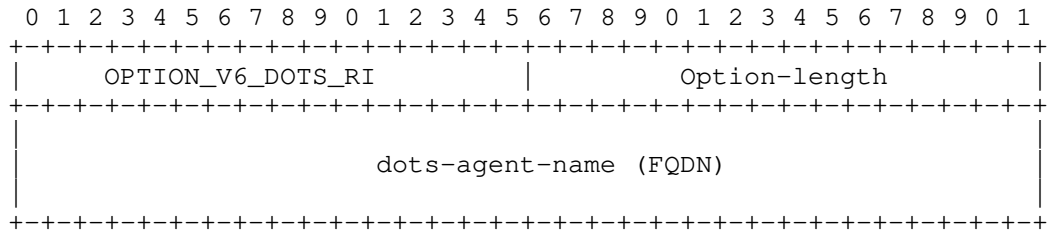


Figure 2: DHCPv6 DOTS Reference Identifier Option

The fields of the option shown in Figure 2 are as follows:

- o Option-code: OPTION_V6_DOTS_RI (TBA1, see Section 9.1)
- o Option-length: Length of the dots-server-name field in octets.
- o dots-agent-name: A fully qualified domain name of the peer DOTS agent. This field is formatted as specified in Section 10 of [RFC8415].

An example of the dots-agent-name encoding is shown in Figure 3. This example conveys the FQDN "dots.example.com".

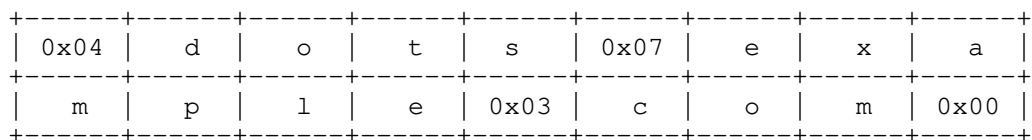


Figure 3: An example of the dots-agent-name Encoding

5.1.2. Format of DOTS Address Option

The DHCPv6 DOTS Address option can be used to configure a list of IPv6 addresses of a DOTS server (or a DOTS client for DOTS Signal Channel Call Home). The format of this option is shown in Figure 4. As a reminder, this format follows the guidelines for creating new DHCPv6 options (Section 5.1 of [RFC7227]).

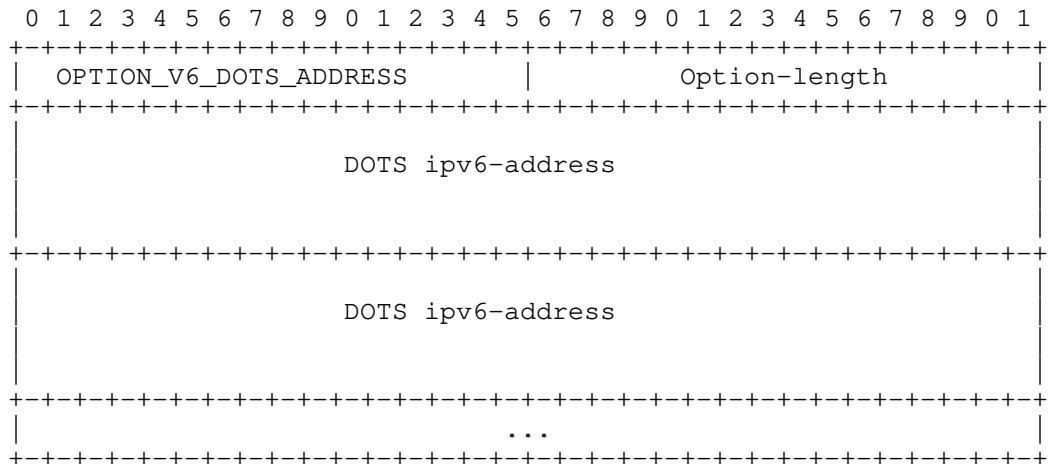


Figure 4: DHCPv6 DOTS Address Option

The fields of the option shown in Figure 4 are as follows:

- o Option-code: OPTION_V6_DOTS_ADDRESS (TBA2, see Section 9.1)
- o Option-length: Length of the 'DOTS ipv6-address(es)' field in octets. MUST be a multiple of 16.
- o DOTS ipv6-address: Includes one or more IPv6 addresses [RFC4291] of the peer DOTS agent to be used by a DOTS agent for establishing a DOTS session.

Note, IPv4-mapped IPv6 addresses (Section 2.5.5.2 of [RFC4291]) are allowed to be included in this option.

To return more than one DOTS agents to the requesting DHCPv6 client, the DHCPv6 server returns multiple instances of OPTION_V6_DOTS_ADDRESS.

5.1.3. DHCPv6 Client Behavior

DHCP clients MAY request options OPTION_V6_DOTS_RI and OPTION_V6_DOTS_ADDRESS, as defined in [RFC8415], Sections 18.2.1, 18.2.2, 18.2.4, 18.2.5, 18.2.6, and 21.7. As a convenience to the reader, it is mentioned here that the DHCP client includes the requested option codes in the Option Request Option.

If the DHCP client receives more than one instance of OPTION_V6_DOTS_RI (or OPTION_V6_DOTS_ADDRESS) option, it MUST use only the first instance of that option.

If the DHCP client receives both `OPTION_V6_DOTS_RI` and `OPTION_V6_DOTS_ADDRESS`, the content of `OPTION_V6_DOTS_RI` is used as reference identifier for authentication purposes (e.g., PKIX [RFC6125]), while the addresses included in `OPTION_V6_DOTS_ADDRESS` are used to reach the peer DOTS agent. In other words, the name conveyed in `OPTION_V6_DOTS_RI` MUST NOT be passed to underlying resolution library in the presence of `OPTION_V6_DOTS_ADDRESS` in a response.

If the DHCP client receives `OPTION_V6_DOTS_RI` only, but `OPTION_V6_DOTS_RI` option contains more than one name, as distinguished by the presence of multiple root labels, the DHCP client MUST use only the first name. Once the name is validated (Section 8 of [RFC8415]), the name is passed to a name resolution library. Moreover, that name is also used as a reference identifier for authentication purposes.

If the DHCP client receives `OPTION_V6_DOTS_ADDRESS` only, the address(es) included in `OPTION_V6_DOTS_ADDRESS` is used to reach the peer DOTS agent. In addition, these addresses can be used as identifiers for authentication.

The DHCP client MUST silently discard multicast and host loopback addresses [RFC6890] conveyed in `OPTION_V6_DOTS_ADDRESS`.

5.2. DHCPv4 DOTS Options

5.2.1. Format of DOTS Reference Identifier Option

The DHCPv4 DOTS Reference Identifier option is used to configure a name of the peer DOTS agent. The format of this option is illustrated in Figure 5.

Code	Length	Peer DOTS agent name					
TBA3	n	s1	s2	s3	s4	s5	...

The values `s1`, `s2`, `s3`, etc. represent the domain name labels in the domain name encoding.

Figure 5: DHCPv4 DOTS Reference Identifier Option

The fields of the option shown in Figure 5 are as follows:

- o Code: `OPTION_V4_DOTS_RI` (TBA3, see Section 9.2);

- o Length: Includes the length of the "DOTS server name" field in octets; the maximum length is 255 octets.
- o Peer DOTS agent name: The domain name of the peer DOTS agent. This field is formatted as specified in Section 10 of [RFC8415].

5.2.2. Format of DOTS Address Option

The DHCPv4 DOTS Address option can be used to configure a list of IPv4 addresses of a peer DOTS agent. The format of this option is illustrated in Figure 6.

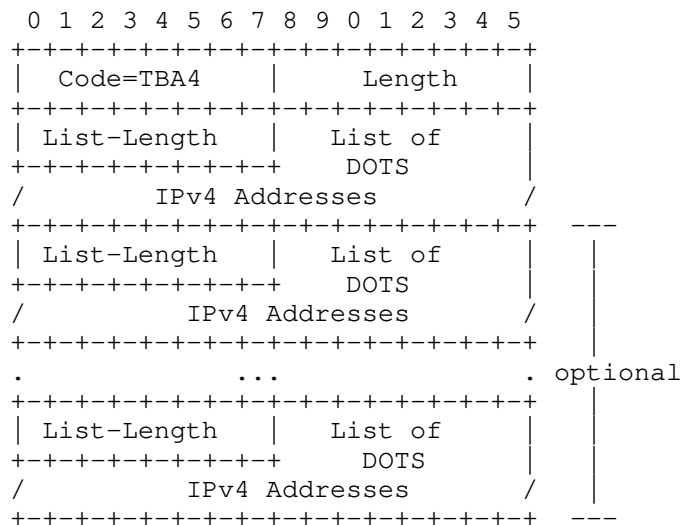
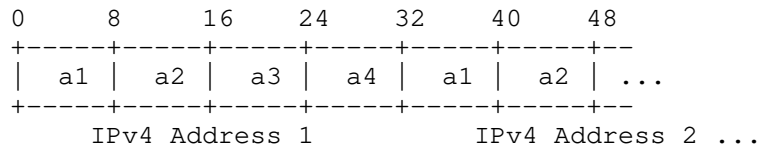


Figure 6: DHCPv4 DOTS Address Option

The fields of the option shown in Figure 6 are as follows:

- o Code: OPTION_V4_DOTS_ADDRESS (TBA4, see Section 9.2);
- o Length: Length of all included data in octets. The minimum length is 5.
- o List-Length: Length of the "List of DOTS IPv4 Addresses" field in octets; MUST be a multiple of 4.
- o List of DOTS IPv4 Addresses: Contains one or more IPv4 addresses of the peer DOTS agent to be used by a DOTS agent. The format of this field is shown in Figure 7.
- o OPTION_V4_DOTS_ADDRESS can include multiple lists of DOTS IPv4 addresses; each list is treated separately as it corresponds to a given peer DOTS agent.

When several lists of DOTS IPv4 addresses are to be included, "List-Length" and "DOTS IPv4 Addresses" fields are repeated.



This format assumes that an IPv4 address is encoded as a1.a2.a3.a4.

Figure 7: Format of the List of DOTS IPv4 Addresses

OPTION_V4_DOTS_ADDRESS is a concatenation-requiring option. As such, the mechanism specified in [RFC3396] MUST be used if OPTION_V4_DOTS_ADDRESS exceeds the maximum DHCPv4 option size of 255 octets.

5.2.3. DHCPv4 Client Behavior

To discover a peer DOTS agent, the DHCPv4 client MUST include both OPTION_V4_DOTS_RI and OPTION_V4_DOTS_ADDRESS in a Parameter Request List Option [RFC2132].

If the DHCP client receives more than one instance of OPTION_V4_DOTS_RI (or OPTION_V4_DOTS_ADDRESS) option, it MUST use only the first instance of that option.

If the DHCP client receives both OPTION_V4_DOTS_RI and OPTION_V4_DOTS_ADDRESS, the content of OPTION_V4_DOTS_RI is used as reference identifier for authentication purposes, while the addresses included in OPTION_V4_DOTS_ADDRESS are used to reach the peer DOTS agent. In other words, the name conveyed in OPTION_V4_DOTS_RI MUST NOT be passed to underlying resolution library in the presence of OPTION_V4_DOTS_ADDRESS in a response.

If the DHCP client receives OPTION_V4_DOTS_RI only, but OPTION_V4_DOTS_RI option contains more than one name, as distinguished by the presence of multiple root labels, the DHCP client MUST use only the first name. Once the name is validated (Section 10 of [RFC8415]), the name is passed to a name resolution library. Moreover, that name is also used as a reference identifier for authentication purposes.

If the DHCP client receives OPTION_V4_DOTS_ADDRESS only, the address(es) included in OPTION_V4_DOTS_ADDRESS is used to reach the peer DOTS server. In addition, these addresses can be used as identifiers for authentication.

The DHCP client MUST silently discard multicast and host loopback addresses conveyed in `OPTION_V4_DOTS_ADDRESS`.

6. Discovery using Service Resolution

This mechanism is performed in two steps:

1. A DNS domain name is retrieved for each combination of interface and address family. A DOTS client has to determine the domain in which it is located relying on dynamic means such as DHCP (Section 5). Implementations MAY allow the user to specify a default name that is used, if no specific name has been configured.
2. Retrieved DNS domain names are then used for S-NAPTR lookups [RFC3958]. Further DNS lookups may be necessary to determine DOTS server IP address(es).

Once the DOTS client has retrieved client's DNS domain or discovered the peer DOTS agent name that needs to be resolved (e.g., Section 5), an S-NAPTR lookup with 'DOTS' application service and the desired protocol tag is made to obtain information necessary to connect to the authoritative DOTS server within the given domain.

This specification defines "DOTS" and "DOTS-CALL-HOME" as application service tags (Sections 9.3.1 and 9.3.2). It also defines "signal.udp" (Section 9.3.3), "signal.tcp" (Section 9.3.4), and "data.tcp" (Section 9.3.5) as application protocol tags. An example is provided in Figure 8.

In the example below, for domain 'example.net', the resolution algorithm will result in IP address(es), port, tag and protocol tuples as follows:

```
example.net.
IN NAPTR 100 10 "" DOTS:signal.udp "" signal.example.net.
IN NAPTR 200 10 "" DOTS:signal.tcp "" signal.example.net.
IN NAPTR 300 10 "" DOTS:data.tcp "" data.example.net.

signal.example.net.
IN NAPTR 100 10 "s" DOTS:signal.udp "" _dots._signal._udp.example.net.
IN NAPTR 200 10 "s" DOTS:signal.tcp "" _dots._signal._tcp.example.net.

data.example.net.
IN NAPTR 100 10 "s" DOTS:data.tcp "" _dots._data._tcp.example.net.

_dots._signal._udp.example.net.
IN SRV 0 0 5000 a.example.net.

_dots._signal._tcp.example.net.
IN SRV 0 0 5001 a.example.net.

_dots._data._tcp.example.net.
IN SRV 0 0 5002 a.example.net.

a.example.net.
IN AAAA 2001:db8::1
```

Order	Protocol	IP address	Port	Tag
1	UDP	2001:db8::1	5000	Signal
2	TCP	2001:db8::1	5001	Signal
3	TCP	2001:db8::1	5002	Data

Figure 8: Sample Example

An example is provided in Figure 9 for the Call Home case.

In the example below, for domain 'example.net', the resolution algorithm will result in IP address(es), port, tag and protocol tuples as follows:

```
example.net.
IN NAPTR 100 10 "" DOTS-CALL-HOME:signal.udp "" signal.example.net.
IN NAPTR 200 10 "" DOTS-CALL-HOME:signal.tcp "" signal.example.net.

signal.example.net.
IN NAPTR 100 10 "s" DOTS-CALL-HOME:signal.udp ""
    _dots-call-home._signal._udp.example.net.
IN NAPTR 200 10 "s" DOTS-CALL-HOME:signal.tcp ""
    _dots-call-home._signal._tcp.example.net.

_dots-call-home._signal._udp.example.net.
IN SRV 0 0 6000 b.example.net.

_dots-call-home._signal._tcp.example.net.
IN SRV 0 0 6001 b.example.net.

b.example.net.
IN AAAA 2001:db8::2
```

Order	Protocol	IP address	Port	Tag
1	UDP	2001:db8::2	6000	Signal
2	TCP	2001:db8::2	6001	Signal

Figure 9: Sample Example for DOTS Signal Channel Call Home

If no DOTS-specific S-NAPTR records can be retrieved, the discovery procedure fails for this domain name (and the corresponding interface and IP protocol version). If more domain names are known, the discovery procedure MAY perform the corresponding S-NAPTR lookups immediately. However, before retrying a lookup that has failed, a DOTS client MUST wait a time period that is appropriate for the encountered error (e.g., NXDOMAIN, timeout, etc.).

7. DNS Service Discovery

DNS-based Service Discovery (DNS-SD) [RFC6763] provides generic solutions for discovering services. DNS-SD defines a set of naming rules for certain DNS record types that they use for advertising and discovering services.

Section 4.1 of [RFC6763] specifies that a service instance name in DNS-SD has the following structure:

<Instance> . <Service> . <Domain>

The <Domain> portion specifies the DNS sub-domain where the service instance is registered. It may be "local.", indicating the mDNS local domain, or it may be a conventional domain name such as "example.com."

The <Service> portion of the DOTS service instance name MUST be "_dots._signal._udp" or "_dots._signal._tcp" or "_dots._data._tcp" or "_dots-call-home._signal._udp" or "_dots-call-home._signal._tcp".

8. Security Considerations

DOTS-related security considerations are discussed in Section 4 of [I-D.ietf-dots-architecture] is to be considered. DOTS agents must authenticate each other using (D)TLS before a DOTS session is considered valid according to the [I-D.ietf-dots-signal-channel].

8.1. DHCP

The security considerations in [RFC2131] and [RFC8415] are to be considered.

8.2. Service Resolution

The primary attack against the methods described in Section 6 is one that would lead to impersonation of a DOTS server. An attacker could attempt to compromise the S-NAPTR resolution. The use of mutual authentication makes it difficult to redirect a DOTS client to an illegitimate DOTS server.

8.3. DNS Service Discovery

Since DNS-SD is just a specification for how to name and use records in the existing DNS system, it has no specific additional security requirements over and above those that already apply to DNS queries and DNS updates. For DNS queries, DNS Security Extensions (DNSSEC) [RFC4033] SHOULD be used where the authenticity of information is important. For DNS updates, secure updates [RFC2136][RFC3007] SHOULD generally be used to control which clients have permission to update DNS records.

9. IANA Considerations

IANA is requested to allocate the SRV service name of "_dots._signal" for DOTS signal channel over UDP or TCP, and the service name of "_dots._data" for DOTS data channel over TCP.

9.1. DHCPv6 Option

IANA is requested to assign the following new DHCPv6 Option Code in the registry maintained in: <http://www.iana.org/assignments/dhcpv6-parameters>.

Value	Description	Client ORO	Singleton Option
TBD1	OPTION_V6_DOTS_RI	Yes	Yes
TBD2	OPTION_V6_DOTS_ADDRESS	Yes	No

9.2. DHCPv4 Option

IANA is requested to assign the following new DHCPv4 Option Code in the registry maintained in: <http://www.iana.org/assignments/bootp-dhcp-parameters/>.

Option Name	Value	Data length	Meaning
OPTION_V4_DOTS_RI	TBA3	Variable; the maximum length is 255 octets.	Includes the name of the DOTS server.
OPTION_V4_DOTS_ADDRESS	TBA4	Variable; the minimum length is 5.	Includes one or multiple lists of DOTS IP addresses; each list is treated as a separate DOTS server.

9.3. Application Service & Application Protocol Tags

This document requests IANA to make the following allocations from the registry available at: <https://www.iana.org/assignments/s-naptr-parameters/s-naptr-parameters.xhtml>.

9.3.1. DOTS Application Service Tag Registration

- o Application Protocol Tag: DOTS
- o Intended Usage: See Section 6
- o Security Considerations: See Section 8
- o Contact Information: <one of the authors>

9.3.2. DOTS Call Home Application Service Tag Registration

- o Application Protocol Tag: DOTS-CALL-HOME
- o Intended Usage: See Section 6
- o Security Considerations: See Section 8
- o Contact Information: <one of the authors>

9.3.3. signal.udp Application Protocol Tag Registration

- o Application Protocol Tag: signal.udp
- o Intended Usage: See Section 6
- o Security Considerations: See Section 8
- o Contact Information: <one of the authors>

9.3.4. signal.tcp Application Protocol Tag Registration

- o Application Protocol Tag: signal.tcp
- o Intended Usage: See Section 6
- o Security Considerations: See Section 8
- o Contact Information: <one of the authors>

9.3.5. data.tcp Application Protocol Tag Registration

- o Application Protocol Tag: data.tcp
- o Intended Usage: See Section 6
- o Security Considerations: See Section 8
- o Contact Information: <one of the authors>

10. Contributors

Prashanth Patil
Cisco Systems, Inc.

Email: praspatti@cisco.com

11. Acknowledgements

Thanks to Brian Carpenter for the review of the BRSKI text.

Many thanks to Russ White for the review, comments, and text contribution.

Thanks for Dan Wing and Pei Wei for the review and comments.

Thanks to Bernie Volz for the review of the DHCP section.

12. References

12.1. Normative References

- [I-D.ietf-dots-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396, DOI 10.17487/RFC3396, November 2002, <<https://www.rfc-editor.org/info/rfc3396>>.
- [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <<https://www.rfc-editor.org/info/rfc3958>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.

- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [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>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

12.2. Informative References

- [I-D.ietf-anima-bootstrapping-keyinfra] Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-21 (work in progress), June 2019.
- [I-D.ietf-dots-architecture] Mortensen, A., K, R., Andreasen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-14 (work in progress), May 2019.
- [I-D.ietf-dots-data-channel] Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-29 (work in progress), May 2019.
- [I-D.ietf-dots-multihoming] Boucadair, M. and R. K, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-01 (work in progress), January 2019.

- [I-D.ietf-dots-signal-call-home]
K, R., Boucadair, M., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home", draft-ietf-dots-signal-call-home-02 (work in progress), May 2019.
- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.
- [I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Secure Zero Touch Provisioning (SZTP)", draft-ietf-netconf-zerotouch-29 (work in progress), January 2019.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<https://www.rfc-editor.org/info/rfc3007>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <<https://www.rfc-editor.org/info/rfc7227>>.

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

DOTS
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

T. Reddy
McAfee
M. Boucadair
Orange
J. Shallow
July 08, 2019

Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
Channel Call Home
draft-ietf-dots-signal-call-home-03

Abstract

This document specifies the DOTS signal channel Call Home service, which enables a DOTS server to initiate a secure connection to a DOTS client, and to receive the attack traffic information from the DOTS client. The DOTS server in turn uses the attack traffic information to identify the compromised devices launching the outgoing DDoS attack and takes appropriate mitigation action(s).

The DOTS Call Home service is not specific to the home networks; the solution targets any deployment which requires to block DDoS attack traffic closer to the source(s) of a DDoS attack.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

- o "This version of this YANG module is part of RFC XXXX;"
- o "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home";
- o "| [RFCXXXX] |"
- o reference: RFC XXXX

Please update this statement with the RFC number to be assigned to the following documents:

- o "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification (used to be I-D.ietf-dots-signal-channel)

Please update TBD statements with the assignment made by IANA to DOTS Signal Channel Call Home.

Also, please update the "revision" date of the YANG module.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. The Problem	3
1.2. The Solution	5
1.3. Applicability Scope	6
2. Terminology	8
3. DOTS Signal Channel Call Home	8
3.1. Procedure	8
3.2. Heartbeat Mechanism	9
3.3. DOTS Signal Channel Extension	10
3.3.1. Mitigation Request	10
3.3.2. Address Sharing Considerations	12
3.3.3. DOTS Signal Call Home YANG Module	15

4.	IANA Considerations	19
4.1.	DOTS Signal Channel Call Home UDP and TCP Port Number . .	19
4.2.	DOTS Signal Channel CBOR Mappings Registry	19
4.3.	New DOTS Conflict Cause	20
4.4.	DOTS Signal Call Home YANG Module	21
5.	Security Considerations	21
6.	Privacy Considerations	22
7.	Contributors	22
8.	Acknowledgements	23
9.	References	23
9.1.	Normative References	23
9.2.	Informative References	24
Appendix A.	Disambiguate Base DOTS Signal vs. Call Home	26
	Authors' Addresses	28

1. Introduction

1.1. The Problem

The DOTS signal channel protocol [I-D.ietf-dots-signal-channel] is used to carry information about a network resource or a network (or a part thereof) that is under a Distributed Denial of Service (DDoS) attack. Such information is sent by a DOTS client to one or multiple DOTS servers so that appropriate mitigation actions are undertaken on traffic deemed suspicious. Various use cases are discussed in [I-D.ietf-dots-use-cases].

Internet of Things (IoT) devices are becoming more and more prevalent in home networks, and with compute and memory becoming cheaper and cheaper, various types of IoT devices become available in the consumer market at affordable prices. But on the downside, the main threat being most of these IoT devices are bought off-the-shelf and most manufacturers haven't considered security in the product design. IoT devices deployed in home networks can be easily compromised, they do not have an easy mechanism to upgrade, and IoT manufacturers may cease manufacture and/or discontinue patching vulnerabilities on IoT devices (Sections 5.4 and 5.5 of [RFC8576]). However, these vulnerable and compromised devices will continue to be used for a long period of time in the home, and the end-user does not know that IoT devices in his/her home are compromised. The compromised IoT devices are typically used for launching DDoS attacks (Section 3 of [RFC8576]) on victims while the owner/administrator of the home network is not aware about such misbehaviors. Similar to other DDoS attacks, the victim in this attack can be an application server, a host, a router, a firewall, or an entire network.

Nowadays, network devices in a home network offer network security (e.g., firewall or Intrusion Protection System (IPS) service on a

home router) to protect the devices connected to the home network from both external and internal attacks. Over the years several techniques have been identified to detect DDoS attacks, some of these techniques can be enabled on home network devices but most of them are used in the Internet Service Provider (ISP)'s network. The ISP offering DDoS mitigation service can detect outgoing DDoS attack traffic originating from its subscribers or the ISP may receive filtering rules (e.g., using BGP flowspec [RFC5575]) from a downstream service provider to filter, block, or rate-limit DDoS attack traffic originating from the ISP's subscribers to a downstream target.

Some of the DDoS attacks like spoofed RST or FIN packets, Slowloris, and Transport Layer Security (TLS) re-negotiation are difficult to detect on a home network device without adversely affecting its performance. The reason is typically home devices such as home routers have fast path to boost the throughput. For every new TCP/UDP flow, only the first few packets are punted through the slow path. Hence, it is not possible to detect various DDoS attacks in the slow path, since the attack payload is sent to the target server after the flow is switched to fast path. Deep Packet Inspection (DPI) of all the packets of a flow would be able to detect some of the attacks. However, a full-fledged DPI to detect these type of DDoS attacks is functionally or operationally not possible for all the devices attached to the home network owing to the memory and CPU limitations of the home routers. Further, for certain DDoS attacks the ability to distinguish legitimate traffic from attacker traffic on a per packet basis is complex. This complexity is due to that the packet itself may look "legitimate" and no attack signature can be identified. The anomaly can be identified only after detailed statistical analysis.

The ISP on the other hand can detect some DDoS attacks originating from a home network (e.g., Section 2.6 of [RFC8517]), but the ISP does not have a mechanism to detect which device in the home network is generating the DDoS attack traffic. The primary reason being that devices in an IPv4 home network are typically behind a Network Address Translation (NAT) border. Even in case of an IPv6 home network, although the ISP can identify the infected device in the home network launching the DDoS traffic by tracking its unique IPv6 address, the infected device can easily change its IPv6 address to evade remediation.

Existing approaches are still suffering from misused access network resources by abusing devices; the support of means for blocking such attacks close to the sources are missing. In particular, the DOTS signal protocol does not discuss cooperative DDoS mitigation between

the network hosting an attack source and the ISP to the suppress the outbound DDoS attack traffic originating from that network.

1.2. The Solution

This specification addresses the problems discussed in Section 1.1 and presents the DOTS signal channel Call Home extension, which enables the DOTS server to initiate a secure connection to the DOTS client, and the DOTS client then conveys the attack traffic information to the DOTS server.

A DOTS client relies upon a variety of triggers to make use of the Call Home function (e.g., scrubbing the traffic from the attack source, receiving an alert from an attack target, a peer DDoS Mitigation System (DMS), or a transit provider). The definition of these triggers is deployment-specific. It is therefore out of the scope of this document to elaborate on how these triggers are made available to a DOTS client.

In a typical deployment scenario, the DOTS server is enabled on a Customer Premises Equipment (CPE), which is aligned with recent trends to enrich the CPE with advanced security features. Unlike classic DOTS deployments [I-D.ietf-dots-use-cases], such DOTS server maintains a single DOTS signal channel session for each DOTS-capable upstream provisioning domain [I-D.ietf-dots-multihoming].

For instance, the DOTS server in the home network initiates the Call Home in 'idle' time and then subsequently the DOTS client in the ISP environment can initiate a mitigation request whenever the ISP detects there is an attack from a compromised device in the DOTS server domain (i.e., from within the home network).

The DOTS server uses the DDoS attack traffic information to identify the compromised device in its domain that is responsible for launching the DDoS attack, optionally notifies a network administrator, and takes appropriate mitigation action(s). A mitigation action can be to quarantine the compromised device or block its traffic to the attack target(s) until the mitigation request is withdrawn.

Other motivations for introducing the Call Home function are discussed in Section 1.1 of [RFC8071].

This document assumes that DOTS servers are provisioned with a way to know how to reach the upstream DOTS client(s), which could occur by a variety of means (e.g., [I-D.ietf-dots-server-discovery]). The specification of such means are out of scope of this document.

1.3. Applicability Scope

The aforementioned problems may be encountered in other deployments than those discussed in Section 1.1 (e.g., data centers, enterprise networks). The solution specified in this document can be used for those deployments to block DDoS attack traffic closer to the source(s) of the attack. The Call Home reference architecture is shown in Figure 1.

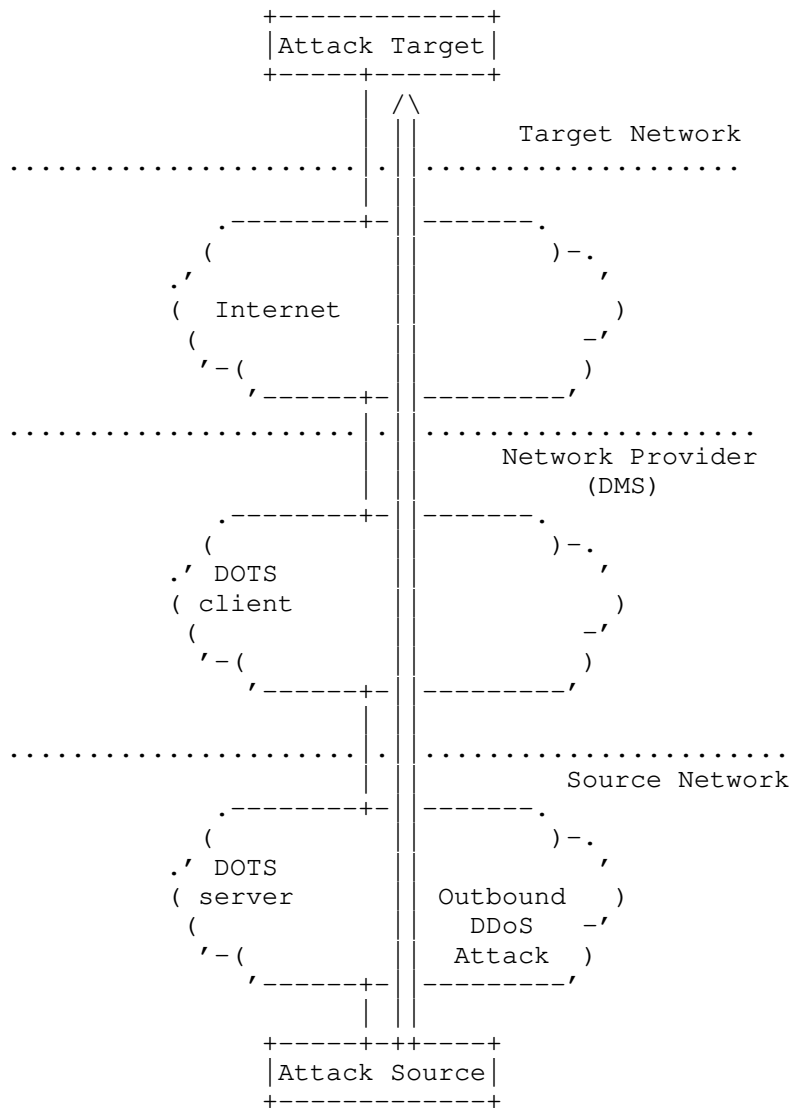


Figure 1: Call Home Reference Architecture

It is out of the scope of this document to identify an exhaustive list of such deployments.

2. Terminology

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

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

The meaning of the symbols in YANG tree diagrams is defined in [RFC8340].

(D)TLS is used for statements that apply to both Transport Layer Security (TLS) [RFC8446] and Datagram Transport Layer Security (DTLS) [RFC6347]. Specific terms are used for any statement that applies to either protocol alone.

3. DOTS Signal Channel Call Home

3.1. Procedure

The DOTS signal channel Call Home extension preserves all but one of the DOTS client/server roles in the DOTS protocol stack, as compared to DOTS client-initiated DOTS signal channel protocol [I-D.ietf-dots-signal-channel]. The role reversal that occurs is at the (D)TLS layer; that is, (1) the DOTS server acts as a DTLS client and the DOTS client acts as a DTLS server or (2) the DOTS server acts as a TLS client initiating the underlying TCP connection and the DOTS client acts as a TLS server. The DOTS server initiates (D)TLS handshake to the DOTS client.

For example, a home network element (e.g., home router) co-located with a DOTS server (likely, a client-domain DOTS gateway) is the (D)TLS server. However, when calling home, the DOTS server initially assumes the role of the (D)TLS client, but the network element's role as a DOTS server remains the same. Furthermore, existing certificate chains and mutual authentication mechanisms between the DOTS agents are unaffected by the Call Home function. This Call Home function enables the DOTS server co-located with a network element (possibly behind NATs and firewalls) reachable by only the intended DOTS client and hence the DOTS server cannot be subjected to DDoS attacks.

Figure 2 illustrates a sample Call Home flow exchange:

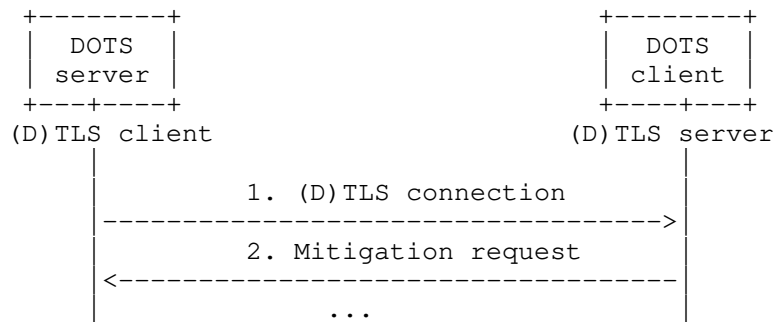


Figure 2: DOTS Signal Channel Call Home Sequence Diagram

The DOTS signal channel Call Home procedure is as follows:

1. If UDP transport is used, the DOTS server begins by initiating a DTLS connection to the DOTS client.

If TCP is used, the DOTS server begins by initiating a TCP connection to the DOTS client. Using this TCP connection, the DOTS server initiates a TLS connection to the DOTS client.

In some cases, a DOTS client and server may have mutual agreement to use a specific port number, such as by explicit configuration or dynamic discovery [I-D.ietf-dots-server-discovery]. Absent such mutual agreement, the DOTS signal channel call home MUST run over port number TBD (that is, DOTS clients must support accepting DTLS (or TCP) connections on TBD) as defined in Section 4.1, for both UDP and TCP. The interaction between the base DOTS signal channel and the call home is discussed in Appendix A.

The Happy Eyeballs mechanism explained in Section 4.3 of [I-D.ietf-dots-signal-channel] can be used for initiating (D)TLS connections.

2. Using this (D)TLS connection, the DOTS client may request, withdraw, or retrieve the status of mitigation requests.

3.2. Heartbeat Mechanism

The Heartbeat mechanism used for the Call Home deviates from the one defined in Section 4.7 of [I-D.ietf-dots-signal-channel]. This section specifies the behavior to be followed by DOTS agents for the Call Home.

Once the (D)TLS section is established between the DOTS agents, the DOTS client contacts the DOTS server to retrieve the session configuration parameters (Section 4.5 of [I-D.ietf-dots-signal-channel]). The DOTS server adjusts the 'heartbeat-interval' to accommodate binding timers used by on-path NATs and firewalls. Heartbeats will be then exchanged by the DOTS agents following the instructions retrieved using the signal channel session configuration exchange.

It is the responsibility of DOTS servers to ensure that on-path translators/firewalls are maintaining a binding so that the same external IP address and/or port number is retained for the DOTS signal channel session. A DOTS client MAY trigger their heartbeat requests immediately after receiving heartbeat probes from its peer DOTS server.

When an outgoing attack that saturates the outgoing link from the DOTS server is detected and reported by a DOTS client, the latter MUST continue to use the signal channel even if no traffic is received from the DOTS server.

If the DOTS server receives traffic from the DOTS client, the DOTS server MUST continue to use the signal channel even if the missing heartbeat allowed threshold is reached.

If the DOTS server does not receive any traffic from the peer DOTS client, the DOTS server sends heartbeat requests to the DOTS client and after maximum 'missing-hb-allowed' threshold is reached, the DOTS server concludes the session is disconnected. Then, the DOTS server MUST try to resume the (D)TLS session.

3.3. DOTS Signal Channel Extension

3.3.1. Mitigation Request

This specification extends the mitigation request defined in Section 4.4.1 of [I-D.ietf-dots-signal-channel] to convey the attacker source prefixes and source port numbers. The DOTS client conveys the following new parameters in the CBOR body of the mitigation request:

source-prefix: A list of attacker prefixes used to attack the target. Prefixes are represented using Classless Inter-Domain Routing (CIDR) notation [RFC4632].

As a reminder, the prefix length MUST be less than or equal to 32 (or 128) for IPv4 (or IPv6).

The prefix list MUST NOT include broadcast, loopback, or multicast addresses. These addresses are considered as invalid values. In addition, the DOTS client MUST validate that attacker prefixes are within the scope of the DOTS server domain.

This is an optional attribute for the base DOTS signal channel operations [I-D.ietf-dots-signal-channel].

source-port-range: A list of port numbers used by the attack traffic flows.

A port range is defined by two bounds, a lower port number (lower-port) and an upper port number (upper-port). When only 'lower-port' is present, it represents a single port number.

For TCP, UDP, Stream Control Transmission Protocol (SCTP) [RFC4960], or Datagram Congestion Control Protocol (DCCP) [RFC4340], a range of ports can be any subrange of 0-65535, for example, 0-1023, 1024-65535, or 1024-49151.

This is an optional attribute for the base DOTS signal channel operations [I-D.ietf-dots-signal-channel].

source-icmp-type-range: A list of ICMP types used by the attack traffic flows. An ICMP type range is defined by two bounds, a lower ICMP type (lower-type) and an upper ICMP type (upper-type). When only 'lower-type' is present, it represents a single ICMP type.

This is an optional attribute for the base DOTS signal channel operations [I-D.ietf-dots-signal-channel].

The 'source-prefix' parameter is a mandatory attribute when the attack traffic information is signaled by a DOTS client in the Call Home scenario (depicted in Figure 2). The 'target-uri' or 'target-fqdn' parameters can be included in a mitigation request for diagnostic purposes to notify the DOTS server domain administrator, but SHOULD NOT be used to determine the target IP addresses. Note that 'target-prefix' becomes a mandatory attribute in the mitigation request signaling the attack information because 'target-uri' and 'target-fqdn' are optional attributes and 'alias-name' will not be conveyed in a mitigation request.

In order to help attack source identification by a DOTS server, the DOTS client SHOULD include in its mitigation request additional information such as 'source-port-range' or 'source-icmp-type-range'. The DOTS client may not include such information if 'source-prefix' conveys an IPv6 address/prefix.

Only immediate mitigation requests (i.e., 'trigger-mitigation' set to 'true') are allowed; DOTS clients MUST NOT send requests with 'trigger-mitigation' set to 'false'. Such requests MUST be discarded by the DOTS server with a 4.00 (Bad Request).

The DOTS server MUST check that the 'source-prefix' is within the scope of the DOTS server domain in the Call Home scenario. Note that in such scenario, the DOTS server considers, by default, that any routeable IP prefix enclosed in 'target-prefix' is within the scope of the DOTS client. Invalid mitigation requests are handled as per Section 4.4.1 of [I-D.ietf-dots-signal-channel].

The DOTS server domain administrator consent MAY be required to block the traffic from the compromised device to the attack target. An implementation MAY have a configuration knob to block the traffic from the compromised device to the attack target with or without DOTS server domain administrator consent. If the attack traffic is blocked, the DOTS server informs the DOTS client that the attack is being mitigated.

If the attack traffic information is identified by the DOTS server or the DOTS server domain administrator as legitimate traffic, the mitigation request is rejected, and 4.09 (Conflict) is returned to the DOTS client. The conflict-clause (defined in Section 4.4.1 of [I-D.ietf-dots-signal-channel]) indicates the cause of the conflict. The following new value is defined:

4: Mitigation request rejected. This code is returned by the DOTS server to indicate the attack traffic has been classified as legitimate traffic.

Once the request is validated by the DOTS server, appropriate actions are enforced to block the attack traffic within the source network. The DOTS client is informed about the progress of the attack mitigation following the rules in [I-D.ietf-dots-signal-channel]. For example, if the DOTS server is embedded in a CPE, it can program the packet processor to punt all the traffic from the compromised device to the target to slow path. The CPE inspects the punted slow path traffic to detect and block the outgoing DDoS attack traffic or quarantine the device (e.g., using MAC level filtering) until it is remediated, and notifies the CPE administrator about the compromised device.

3.3.2. Address Sharing Considerations

If a Carrier Grade NAT (CGN, including NAT64) is located between the DOTS client domain and DOTS server domain, communicating an external IP address in a mitigation request is likely to be discarded by the

DOTS server because the external IP address is not visible locally to the DOTS server (see Figure 3). The DOTS server is only aware of the internal IP addresses/prefixes bound to its domain. Thus, the DOTS client MUST NOT include the external IP address and/or port number identifying the suspect attack source, but MUST include the internal IP address and/or port number. To that aim, the DOTS client SHOULD rely on mechanisms, such as [RFC8512] or [RFC8513], to retrieve the internal IP address and port number which are mapped to an external IP address and port number.

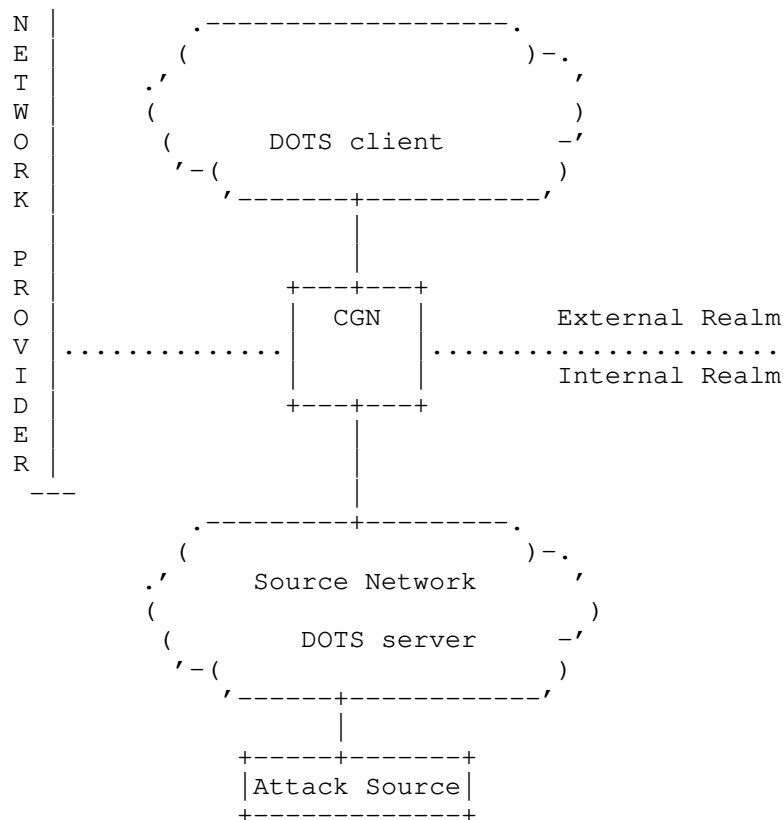


Figure 3: Example of a CGN between DOTS Domains

If a MAP Border Relay [RFC7597] or lwAFTR [RFC7596] is enabled in the provider's domain to service its customers, the identification of an attack source bound to an IPv4 address/prefix MUST also rely on source port numbers because the same IPv4 address is assigned to multiple customers. The port information is required to unambiguously identify the source of an attack.

If a translator is enabled on the boundaries of the domain hosting the DOTS server (e.g., a CPE with NAT enabled as shown in Figures 4 and 5), the DOTS server uses the attack traffic information conveyed in a mitigation request to find the internal source IP address of the compromised device and blocks the traffic from the compromised device traffic to the attack target until the mitigation request is withdrawn. Doing so allows to isolate the suspicious device while avoiding to disturb other services.

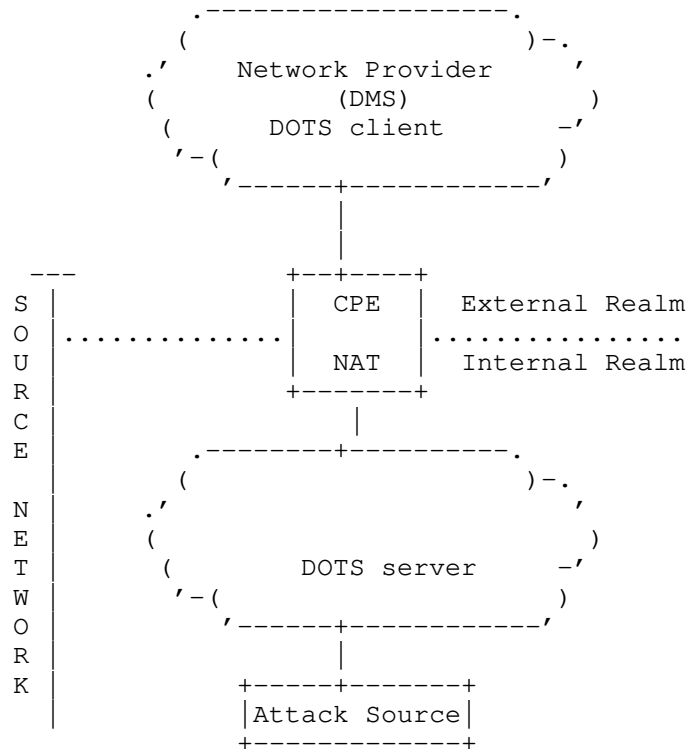


Figure 4: Example of a DOTS Server Domain with a NAT Embedded in a CPE

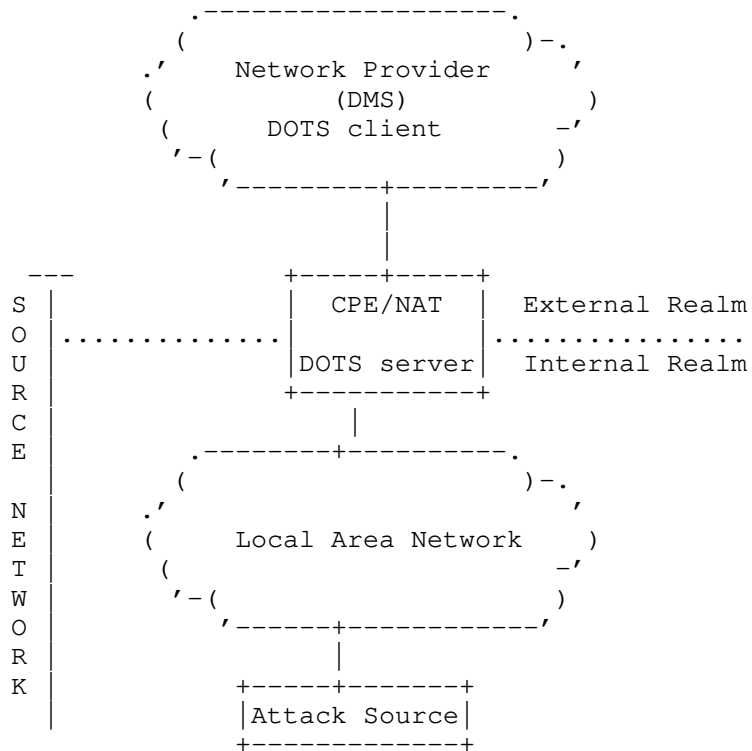


Figure 5: Example of a DOTS Server and a NAT Embedded in a CPE

3.3.3. DOTS Signal Call Home YANG Module

3.3.3.1. Tree Structure

This document augments the "dots-signal-channel" DOTS signal YANG module defined in [I-D.ietf-dots-signal-channel] for signaling the attack traffic information. This document defines the YANG module "ietf-dots-call-home", which has the following tree structure:

```

module: ietf-dots-call-home
  augment /ietf-signal:dots-signal/ietf-signal:message-type
    /ietf-signal:mitigation-scope/ietf-signal:scope:
    +--rw source-prefix*      inet:ip-prefix {source-signaling}?
    +--rw source-port-range* [lower-port] {source-signaling}?
    |   +--rw lower-port      inet:port-number
    |   +--rw upper-port?     inet:port-number
    +--rw source-icmp-type-range*
    |   [lower-type] {source-signaling}?
    +--rw lower-type          uint8
    +--rw upper-type?         uint8

```

3.3.3.2. YANG Module

This module uses the common YANG types defined in [RFC6991].

<CODE BEGINS> file "ietf-dots-call-home@2019-04-25.yang"

```

module ietf-dots-call-home {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-call-home";
  prefix call-home;

  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-dots-signal-channel {
    prefix ietf-signal;
    reference
      "RFC YYYY: Distributed Denial-of-Service Open Threat
      Signaling (DOTS) Signal 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:   Konda, Tirumaleswar Reddy
              <mailto:TirumaleswarReddy_Konda@McAfee.com>;

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

```

Author: Jon Shallow
<mailto:ietf-supjps@jpshallow.com>;

description

"This module contains YANG definitions for the signaling messages exchanged between a DOTS client and a DOTS server for the Call Home deployment scenario.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

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

revision 2019-04-25 {

description

"Initial revision.";

reference

"RFC XXXX: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Call Home";

}

feature source-signaling {

description

"This feature means that source-related information can be supplied in mitigation requests.";

}

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

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

if-feature source-signaling;

description "Attacker source details.";

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

4. IANA Considerations

4.1. DOTS Signal Channel Call Home UDP and TCP Port Number

IANA is requested to assign the port number TBD to the DOTS signal channel Call Home protocol for both UDP and TCP from the "Service Name and Transport Protocol Port Number Registry" available at: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

Service Name:	dots-call-home
Port Number:	TBD
Transport Protocol(s):	TCP/UDP
Description:	DOTS Signal Channel Call Home
Assignee:	IESG <iesg@ietf.org>
Contact:	IETF Chair <chair@ietf.org>
Reference:	RFC XXXX

The assignment of port number 4647 is strongly suggested (DOTS signal channel uses port number 4646).

4.2. DOTS Signal Channel CBOR Mappings Registry

This specification registers the 'source-prefix', 'source-port-range', and 'source-icmp-type-range' parameters in the IANA "DOTS Signal Channel CBOR Key Values" registry established by [I-D.ietf-dots-signal-channel] (Figure 6).

The 'source-prefix', 'source-port-range', and 'source-icmp-type-range' are comprehension-optional parameters.

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

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
source-prefix	leaf-list inet: ip-prefix	0x8000 (TBD1)	4 array	Array
source-port-range	list	0x8001 (TBD2)	3 text string 4 array	String Array
source-icmp-type-range	list	0x8002 (TBD3)	4 array	Array
lower-type	uint8	0x8003 (TBD4)	0 unsigned	Number
upper-type	uint8	0x8004 (TBD5)	0 unsigned	Number

Figure 6: Assigned DOTS Signal Channel CBOR Key Values

4.3. New DOTS Conflict Cause

This document requests IANA to assign a new code from the "DOTS Signal Channel Conflict Cause Codes" registry:

Code	Label	Description	Reference
4	request-rejected-legitimate-traffic	Mitigation request rejected. This code is returned by the DOTS server to indicate the attack traffic has been classified as legitimate traffic.	[RFCXXXX]

4.4. DOTS Signal Call Home YANG Module

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC7950] within the "YANG Parameters" registry:

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

5. Security Considerations

This document deviates from classic DOTS signal channel usage by having the DOTS server initiate the (D)TLS connection. DOTS signal channel related security considerations discussed in Section 10 of [I-D.ietf-dots-signal-channel] MUST be considered. DOTS agents MUST authenticate each other using (D)TLS before a DOTS signal channel session is considered valid.

An attacker may launch a DoS attack on the DOTS client by having it perform computationally expensive operations, before deducing that the attacker doesn't possess a valid key. For instance, in TLS 1.3 [RFC8446], the ServerHello message contains a Key Share value based on an expensive asymmetric key operation for key establishment. Common precautions mitigating DoS attacks are recommended, such as temporarily blacklisting the source address after a set number of unsuccessful authentication attempts.

DOTS servers may not blindly trust mitigation requests from DOTS clients. For example, DOTS servers can use the attack flow information in a mitigation request to enable full-fledged packet inspection function to inspect all the traffic from the compromised device to the target or to re-direct the traffic from the compromised device to the target to a DDoS mitigation system to scrub the suspicious traffic. DOTS servers can also seek the consent of DOTS server domain administrator to block the traffic from the compromised device to the target (see Section 3.3.1).

6. Privacy Considerations

The considerations discussed in [RFC6973] were taken into account to assess whether the DOTS Call Home extension introduces privacy threats.

Concretely, the protocol does not leak any new information that can be used to ease surveillance. In particular, the DOTS server is not required to share information that is local to its network (e.g., internal identifiers of an attack source) with the DOTS client.

The DOTS Call Home extension does not preclude the validation of mitigation requests received from a DOTS client. For example, a security service running on the CPE may require administrator's consent before the CPE acts upon the mitigation request indicated by the DOTS client. How the consent is obtained is out of scope of this document.

Note that a DOTS server can seek for an administrator's consent, validate the request by inspecting the traffic, or proceed with both.

The DOTS Call Home extension is only advisory in nature. Concretely, the DOTS Call Home extension does not impose any action to be enforced within the home network; it is up to the DOTS server (and/or network administrator) to decide whether and which actions are required.

Moreover, the DOTS Call Home extension avoids misattribution by appropriately identifying the network to which a suspect attack source belongs to (e.g., address sharing issues discussed in Section 3.3.1).

Triggers to send a DOTS mitigation request to a DOTS server are deployment-specific. For example, a DOTS client may rely on the output of some DDoS detection systems deployed within the DOTS client domain to detect potential outbound DDoS attacks or on abuse claims received from remote victim networks. Such DDoS detection and mitigation techniques are not meant to track the activity of users, but to protect the Internet and avoid altering the IP reputation of the DOTS client domain.

7. Contributors

The following individuals have contributed to this document:

Joshi Harsha
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: harsha_joshi@mcafee.com

Wei Pan
Huawei Technologies
China

Email: william.panwei@huawei.com

8. Acknowledgements

Thanks to Wei Pei, Xia Liang, Roman Danyliw, Dan Wing, Toema Gavrichenkov, Daniel Migault, and Valery Smyslov for the comments.

9. References

9.1. Normative References

- [I-D.ietf-dots-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

9.2. Informative References

- [I-D.ietf-dots-multihoming]
Boucadair, M. and R. K., "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-01 (work in progress), January 2019.
- [I-D.ietf-dots-server-discovery]
Boucadair, M. and R. K., "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server Discovery", draft-ietf-dots-server-discovery-04 (work in progress), June 2019.
- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-17 (work in progress), January 2019.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<https://www.rfc-editor.org/info/rfc4340>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<https://www.rfc-editor.org/info/rfc4632>>.
- [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>>.
- [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>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <<https://www.rfc-editor.org/info/rfc7596>>.
- [RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", RFC 7597, DOI 10.17487/RFC7597, July 2015, <<https://www.rfc-editor.org/info/rfc7597>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<https://www.rfc-editor.org/info/rfc8071>>.
- [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>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.
- [RFC8513] Boucadair, M., Jacquenet, C., and S. Sivakumar, "A YANG Data Model for Dual-Stack Lite (DS-Lite)", RFC 8513, DOI 10.17487/RFC8513, January 2019, <<https://www.rfc-editor.org/info/rfc8513>>.

- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.
- [RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <<https://www.rfc-editor.org/info/rfc8576>>.
- [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>>.

Appendix A. Disambiguate Base DOTS Signal vs. Call Home

With the call home extension, there is a chance that two DOTS agents can simultaneously establish two DOTS signal channels with different directions (base DOTS signal channel and DOTS signal channel call home). Here is one example drawn from the home network. Nevertheless, the outcome of the discussion is not specific to these networks, but applies to any DOTS call home scenario.

In the call home scenario, the DOTS server in the home network can mitigate the DDoS attacks launched by the compromised device in its domain by receiving the mitigation request sent by the DOTS client in the ISP environment. In addition, the DOTS client in the home network can initiate a mitigation request to the DOTS server in the ISP environment to ask for help when the home network is under a DDoS attack. Such DOTS server and DOTS client in the home network can co-locate in the same home network element (e.g., the Customer Premises Equipment). In this case, with the same peer at the same time the home network element will have the basic DOTS signal channel defined in [I-D.ietf-dots-signal-channel] and the call home DOTS signal channel defined in this specification. Thus these two signal channels need to be distinguished when they are both supported.

In the call home scenario, the DOTS server in, for example, the home network can mitigate the DDoS attacks launched by the compromised device in its domain by receiving the mitigation request sent by the DOTS client in the ISP environment. In addition, the DOTS client in the home network can initiate a mitigation request to the DOTS server in the ISP environment to ask for help when the home network is under a DDoS attack. Such DOTS server and DOTS client in the home network can co-locate in the same home network element (e.g., the Customer Premises Equipment). In this case, with the same peer at the same

time the home network element will have the basic DOTS signal channel defined in [I-D.ietf-dots-signal-channel] and the call home DOTS signal channel defined in this specification. Thus these two signal channels need to be distinguished when they are both supported. Two approaches have been considered for distinguishing the two DOTS signal channels, but only the one that using the dedicated port number has been chosen as the best choice.

By using a dedicated port number for each, these two signal channels can be separated unambiguously and easily. For example, the CPE uses the port number 4646 defined in [I-D.ietf-dots-signal-channel] to initiate the basic signal channel to the ISP when it acts as the DOTS client, and uses the port number TBD to initiate the call home signal channel. Based on the different ports, the ISP can directly decide which kind of procedures should follow immediately after it receives the DOTS messages. This approach just requires two (D)TLS sessions to be established respectively for the basic signal channel and call home signal channel.

The other approach is signaling the role of each DOTS agent (e.g., by using the DOTS data channel). For example, the DOTS agent in the home network first initiates a DOTS data channel to the peer DOTS agent in the ISP environment, at this time the DOTS agent in the home network is the DOTS client and the peer DOTS agent in the ISP environment is the DOTS server. After that, the DOTS agent in the home network retrieves the DOTS call home capability of the peer DOTS agent. If the peer supports the call home extension, the DOTS agent needs to subscribe to the peer to use this extension. Then the reversal of DOTS role can be recognized as done by both DOTS agents. When the DOTS agent in the ISP environment, which now is the DOTS client, wants to filter the attackers' traffic, it requests the DOTS agent in the home network, which now is the DOTS server, for help.

Signaling the role will complicate the DOTS protocol, and this complexity is not required in context where call home extension is not required or only when call home extension is needed. Besides, the DOTS data channel may not work during attack time. Even if changing the above example from using the DOTS data channel to the DOTS signal channel, the more procedures will still reduce the efficiency. Using the dedicated port number is much easier and more concise compared to the second approach, and its cost that establishing two (D)TLS sessions is much less. So, using a dedicated port number for the call home extension is chosen in this specification.

Authors' Addresses

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

Email: kondtir@gmail.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Jon Shallow
UK

Email: supjps-ietf@jpshallow.com

DOTS
Internet-Draft
Intended status: Standards Track
Expires: January 24, 2020

T. Reddy, Ed.
McAfee
M. Boucadair, Ed.
Orange
P. Patil
Cisco
A. Mortensen
Arbor Networks, Inc.
N. Teague
Iron Mountain Data Centers
July 23, 2019

Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
Channel Specification
draft-ietf-dots-signal-channel-36

Abstract

This document specifies the DOTS signal channel, a protocol for signaling the need for protection against Distributed Denial-of-Service (DDoS) attacks to a server capable of enabling network traffic mitigation on behalf of the requesting client.

A companion document defines the DOTS data channel, a separate reliable communication layer for DOTS management and configuration purposes.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

- o "This version of this YANG module is part of RFC XXXX;"
- o "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification";
- o "| [RFCXXXX] |"
- o reference: RFC XXXX

Please update this statement with the RFC number to be assigned to the following documents:

- o "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification (used to be I-D.ietf-dots-data-channel)"

Please update TBD/TBD1/TBD2 statements with the assignments made by IANA to DOTS Signal Channel Protocol.

Also, please update the "revision" date of the YANG modules.

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 January 24, 2020.

Copyright Notice

Copyright (c) 2019 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	4
2. Terminology	6
3. Design Overview	6
4. DOTS Signal Channel: Messages & Behaviors	9
4.1. DOTS Server(s) Discovery	9
4.2. CoAP URIs	10
4.3. Happy Eyeballs for DOTS Signal Channel	10
4.4. DOTS Mitigation Methods	12
4.4.1. Request Mitigation	13

4.4.2.	Retrieve Information Related to a Mitigation	29
4.4.2.1.	DOTS Servers Sending Mitigation Status	34
4.4.2.2.	DOTS Clients Polling for Mitigation Status	37
4.4.3.	Efficacy Update from DOTS Clients	38
4.4.4.	Withdraw a Mitigation	40
4.5.	DOTS Signal Channel Session Configuration	41
4.5.1.	Discover Configuration Parameters	43
4.5.2.	Convey DOTS Signal Channel Session Configuration . .	47
4.5.3.	Configuration Freshness and Notifications	53
4.5.4.	Delete DOTS Signal Channel Session Configuration . .	54
4.6.	Redirected Signaling	55
4.7.	Heartbeat Mechanism	57
5.	DOTS Signal Channel YANG Modules	58
5.1.	Tree Structure	59
5.2.	IANA DOTS Signal Channel YANG Module	61
5.3.	IETF DOTS Signal Channel YANG Module	65
6.	YANG/JSON Mapping Parameters to CBOR	75
7.	(D)TLS Protocol Profile and Performance Considerations . . .	77
7.1.	(D)TLS Protocol Profile	77
7.2.	(D)TLS 1.3 Considerations	79
7.3.	DTLS MTU and Fragmentation	81
8.	Mutual Authentication of DOTS Agents & Authorization of DOTS Clients	82
9.	IANA Considerations	84
9.1.	DOTS Signal Channel UDP and TCP Port Number	84
9.2.	Well-Known 'dots' URI	84
9.3.	Media Type Registration	84
9.4.	CoAP Content-Formats Registration	85
9.5.	CBOR Tag Registration	85
9.6.	DOTS Signal Channel Protocol Registry	86
9.6.1.	DOTS Signal Channel CBOR Key Values Sub-Registry . .	86
9.6.1.1.	Registration Template	86
9.6.1.2.	Initial Sub-Registry Content	87
9.6.2.	Status Codes Sub-Registry	89
9.6.3.	Conflict Status Codes Sub-Registry	90
9.6.4.	Conflict Cause Codes Sub-Registry	92
9.6.5.	Attack Status Codes Sub-Registry	92
9.7.	DOTS Signal Channel YANG Modules	93
10.	Security Considerations	94
11.	Contributors	96
12.	Acknowledgements	97
13.	References	97
13.1.	Normative References	97
13.2.	Informative References	100
Appendix A.	CUID Generation	105
Authors' Addresses	105

1. Introduction

A distributed denial-of-service (DDoS) attack is a distributed attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale for an effective attack can be achieved by compromising enough end-hosts and using those infected hosts to perpetrate and amplify the attack. The victim in this attack can be an application server, a host, a router, a firewall, or an entire network.

Network applications have finite resources like CPU cycles, the number of processes or threads they can create and use, the maximum number of simultaneous connections they can handle, the limited resources of the control plane, etc. When processing network traffic, such applications are supposed to use these resources to provide the intended functionality in the most efficient manner. However, a DDoS attacker may be able to prevent an application from performing its intended task by making the application exhaust its finite resources.

A TCP DDoS SYN-flood [RFC4987], for example, is a memory-exhausting attack while an ACK-flood is a CPU-exhausting attack. Attacks on the link are carried out by sending enough traffic so that the link becomes congested, thereby likely causing packet loss for legitimate traffic. Stateful firewalls can also be attacked by sending traffic that causes the firewall to maintain an excessive number of states that may jeopardize the firewall's operation overall, besides likely performance impacts. The firewall then runs out of memory, and can no longer instantiate the states required to process legitimate flows. Other possible DDoS attacks are discussed in [RFC4732].

In many cases, it may not be possible for network administrators to determine the cause(s) of an attack. They may instead just realize that certain resources seem to be under attack. This document defines a lightweight protocol that allows a DOTS client to request mitigation from one or more DOTS servers for protection against detected, suspected, or anticipated attacks. This protocol enables cooperation between DOTS agents to permit a highly-automated network defense that is robust, reliable, and secure. Note that "secure" means the support of the features defined in Section 2.4 of [RFC8612].

An example of a network diagram that illustrates a deployment of DOTS agents is shown in Figure 1. In this example, a DOTS server is operating on the access network. A DOTS client is located on the LAN (Local Area Network), while a DOTS gateway is embedded in the CPE (Customer Premises Equipment).

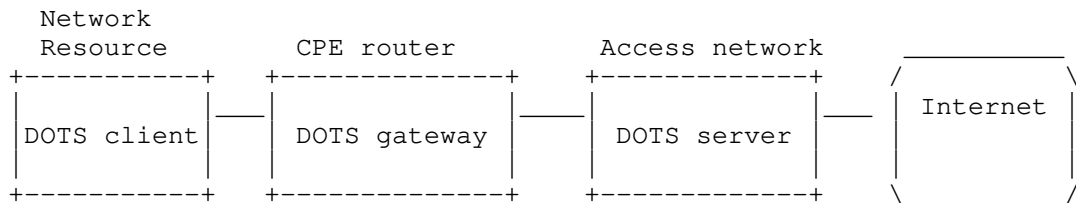


Figure 1: Sample DOTS Deployment (1)

DOTS servers can also be reachable over the Internet, as depicted in Figure 2.

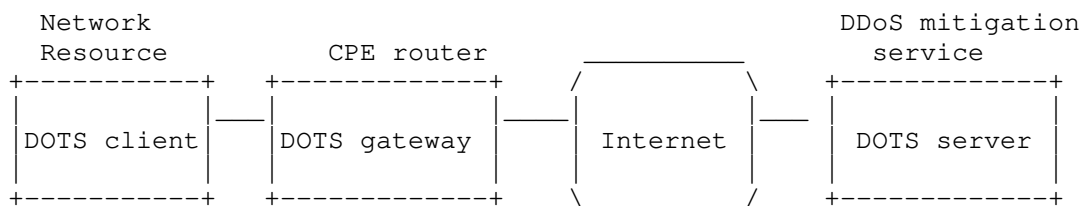


Figure 2: Sample DOTS Deployment (2)

In typical deployments, the DOTS client belongs to a different administrative domain than the DOTS server. For example, the DOTS client is embedded in a firewall protecting services owned and operated by a customer, while the DOTS server is owned and operated by a different administrative entity (service provider, typically) providing DDoS mitigation services. The latter might or might not provide connectivity services to the network hosting the DOTS client.

The DOTS server may (not) be co-located with the DOTS mitigator. In typical deployments, the DOTS server belongs to the same administrative domain as the mitigator. The DOTS client can communicate directly with a DOTS server or indirectly via a DOTS gateway.

The document adheres to the DOTS architecture [I-D.ietf-dots-architecture]. The requirements for DOTS signal channel protocol are documented in [RFC8612]. This document satisfies all the use cases discussed in [I-D.ietf-dots-use-cases].

This document focuses on the DOTS signal channel. This is a companion document of the DOTS data channel specification [I-D.ietf-dots-data-channel] that defines a configuration and a bulk data exchange mechanism supporting 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.

(D)TLS is used for statements that apply to both Transport Layer Security [RFC5246][RFC8446] and Datagram Transport Layer Security [RFC6347]. Specific terms are used for any statement that applies to either protocol alone.

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

The meaning of the symbols in YANG tree diagrams is defined in [RFC8340].

3. Design Overview

The DOTS signal channel is built on top of the Constrained Application Protocol (CoAP) [RFC7252], a lightweight protocol originally designed for constrained devices and networks. The many features of CoAP (expectation of packet loss, support for asynchronous Non-confirmable messaging, congestion control, small message overhead limiting the need for fragmentation, use of minimal resources, and support for (D)TLS) makes it a good candidate to build the DOTS signaling mechanism from.

The DOTS signal channel is layered on existing standards (Figure 3).

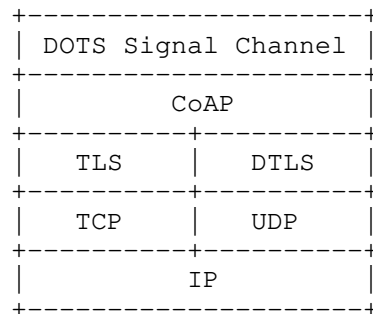


Figure 3: Abstract Layering of DOTS Signal Channel over CoAP over (D)TLS

In some cases, a DOTS client and server may have mutual agreement to use a specific port number, such as by explicit configuration or

dynamic discovery [I-D.ietf-dots-server-discovery]. Absent such mutual agreement, the DOTS signal channel MUST run over port number TBD as defined in Section 9.1, for both UDP and TCP. In order to use a distinct port number (as opposed to TBD), DOTS clients and servers SHOULD support a configurable parameter to supply the port number to use.

Note: The rationale for not using the default port number 5684 ((D)TLS CoAP) is to avoid the discovery of services and resources discussed in [RFC7252] and allow for differentiated behaviors in environments where both a DOTS gateway and an IoT gateway (e.g., Figure 3 of [RFC7452]) are co-located.

Particularly, the use of a default port number is meant to simplify DOTS deployment in scenarios where no explicit IP address configuration is required. For example, the use of the default router as DOTS server aims to ease DOTS deployment within LANs (in which, CPEs embed a DOTS gateway as illustrated in Figures 1 and 2) without requiring a sophisticated discovery method and configuration tasks within the LAN. The use of anycast is meant to simplify DOTS client configuration, including service discovery. In such anycast-based scenario, a DOTS client initiating a DOTS session to the DOTS server anycast address may, for example, be (1) redirected to the DOTS server unicast address to be used by the DOTS client following the procedure discussed in Section 4.6 or (2) relayed to a unicast DOTS server.

The signal channel uses the "coaps" URI scheme defined in Section 6 of [RFC7252] and the "coaps+tcp" URI scheme defined in Section 8.2 of [RFC8323] to identify DOTS server resources accessible using CoAP over UDP secured with DTLS and CoAP over TCP secured with TLS, respectively.

The DOTS signal channel can be established between two DOTS agents prior or during an attack. The DOTS signal channel is initiated by the DOTS client. The DOTS client can then negotiate, configure, and retrieve the DOTS signal channel session behavior with its DOTS peer (Section 4.5). Once the signal channel is established, the DOTS agents may periodically send heartbeats to keep the channel active (Section 4.7). At any time, the DOTS client may send a mitigation request message (Section 4.4) to a DOTS server over the active signal channel. While mitigation is active (because of the higher likelihood of packet loss during a DDoS attack), the DOTS server periodically sends status messages to the client, including basic mitigation feedback details. Mitigation remains active until the DOTS client explicitly terminates mitigation, or the mitigation lifetime expires. Also, the DOTS server may rely on the signal

channel session loss to trigger mitigation for pre-configured mitigation requests (if any).

DOTS signaling can happen with DTLS over UDP and TLS over TCP. Likewise, DOTS requests may be sent using IPv4 or IPv6 transfer capabilities. A Happy Eyeballs procedure for DOTS signal channel is specified in Section 4.3.

A DOTS client is entitled to access only to resources it creates. In particular, a DOTS client can not retrieve data related to mitigation requests created by other DOTS clients of the same DOTS client domain.

Messages exchanged between DOTS agents are serialized using Concise Binary Object Representation (CBOR) [RFC7049], a binary encoding scheme designed for small code and message size. CBOR-encoded payloads are used to carry signal channel-specific payload messages which convey request parameters and response information such as errors. In order to allow reusing data models across protocols, [RFC7951] specifies the JavaScript Object Notation (JSON) encoding of YANG-modeled data. A similar effort for CBOR is defined in [I-D.ietf-core-yang-cbor].

DOTS agents determine that a CBOR data structure is a DOTS signal channel object from the application context, such as from the port number assigned to the DOTS signal channel. The other method DOTS agents use to indicate that a CBOR data structure is a DOTS signal channel object is the use of the "application/dots+cbor" content type (Section 9.3).

This document specifies a YANG module for representing DOTS mitigation scopes, DOTS signal channel session configuration data, and DOTS redirected signaling (Section 5). All parameters in the payload of the DOTS signal channel are mapped to CBOR types as specified in Table 4 (Section 6).

In order to prevent fragmentation, DOTS agents must follow the recommendations documented in Section 4.6 of [RFC7252]. Refer to Section 7.3 for more details.

DOTS agents MUST support GET, PUT, and DELETE CoAP methods. The payload included in CoAP responses with 2.xx Response Codes MUST be of content type "application/dots+cbor". CoAP responses with 4.xx and 5.xx error Response Codes MUST include a diagnostic payload (Section 5.5.2 of [RFC7252]). The Diagnostic Payload may contain additional information to aid troubleshooting.

In deployments where multiple DOTS clients are enabled in a network (owned and operated by the same entity), the DOTS server may detect conflicting mitigation requests from these clients. This document does not aim to specify a comprehensive list of conditions under which a DOTS server will characterize two mitigation requests from distinct DOTS clients as conflicting, nor recommend a DOTS server behavior for processing conflicting mitigation requests. Those considerations are implementation- and deployment-specific. Nevertheless, the document specifies the mechanisms to notify DOTS clients when conflicts occur, including the conflict cause (Section 4.4).

In deployments where one or more translators (e.g., Traditional NAT [RFC3022], CGN [RFC6888], NAT64 [RFC6146], NPTv6 [RFC6296]) are enabled between the client's network and the DOTS server, DOTS signal channel messages forwarded to a DOTS server MUST NOT include internal IP addresses/prefixes and/or port numbers; external addresses/prefixes and/or port numbers as assigned by the translator MUST be used instead. This document does not make any recommendation about possible translator discovery mechanisms. The following are some (non-exhaustive) deployment examples that may be considered:

- o Port Control Protocol (PCP) [RFC6887] or Session Traversal Utilities for NAT (STUN) [RFC5389] may be used to retrieve the external addresses/prefixes and/or port numbers. Information retrieved by means of PCP or STUN will be used to feed the DOTS signal channel messages that will be sent to a DOTS server.
- o A DOTS gateway may be co-located with the translator. The DOTS gateway will need to update the DOTS messages, based upon the local translator's binding table.

4. DOTS Signal Channel: Messages & Behaviors

4.1. DOTS Server(s) Discovery

This document assumes that DOTS clients are provisioned with the reachability information of their DOTS server(s) using any of a variety of means (e.g., local configuration, or dynamic means such as DHCP [I-D.ietf-dots-server-discovery]). The description of such means is out of scope of this document.

Likewise, it is out of scope of this document to specify the behavior to be followed by a DOTS client to send DOTS requests when multiple DOTS servers are provisioned (e.g., contact all DOTS servers, select one DOTS server among the list). Such behavior is specified in other documents (e.g., [I-D.ietf-dots-multhoming]).

4.2. CoAP URIs

The DOTS server MUST support the use of the path-prefix of `"/.well-known/"` as defined in [RFC5785] and the registered name of `"dots"`. Each DOTS operation is indicated by a path-suffix that indicates the intended operation. The operation path (Table 1) is appended to the path-prefix to form the URI used with a CoAP request to perform the desired DOTS operation.

Operation	Operation Path	Details
Mitigation	/mitigate	Section 4.4
Session configuration	/config	Section 4.5

Table 1: Operations and their Corresponding URIs

4.3. Happy Eyeballs for DOTS Signal Channel

[RFC8612] mentions that DOTS agents will have to support both connectionless and connection-oriented protocols. As such, the DOTS signal channel is designed to operate with DTLS over UDP and TLS over TCP. Further, a DOTS client may acquire a list of IPv4 and IPv6 addresses (Section 4.1), each of which can be used to contact the DOTS server using UDP and TCP. If no list of IPv4 and IPv6 addresses to contact the DOTS server is configured (or discovered), the DOTS client adds the IPv4/IPv6 addresses of its default router to the candidate list to contact the DOTS server.

The following specifies the procedure to follow to select the address family and the transport protocol for sending DOTS signal channel messages.

Such procedure is needed to avoid experiencing long connection delays. For example, if an IPv4 path to reach a DOTS server is functional, but the DOTS server's IPv6 path is non-functional, a dual-stack DOTS client may experience a significant connection delay compared to an IPv4-only DOTS client, in the same network conditions. The other problem is that if a middlebox between the DOTS client and DOTS server is configured to block UDP traffic, the DOTS client will fail to establish a DTLS association with the DOTS server and, as a consequence, will have to fall back to TLS over TCP, thereby incurring significant connection delays.

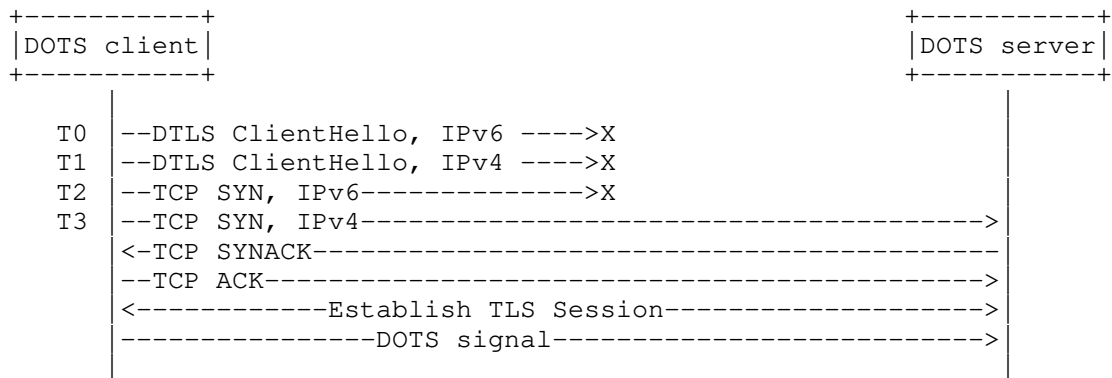
To overcome these connection setup problems, the DOTS client attempts to connect to its DOTS server(s) using both IPv6 and IPv4, and tries

both DTLS over UDP and TLS over TCP following a DOTS Happy Eyeballs approach. To some extent, this approach is similar to the Happy Eyeballs mechanism defined in [RFC8305]. The connection attempts are performed by the DOTS client when it initializes, or in general when it has to select an address family and transport to contact its DOTS server. The results of the Happy Eyeballs procedure are used by the DOTS client for sending its subsequent messages to the DOTS server. The difference in behavior with respect to the Happy Eyeballs mechanism [RFC8305] are listed below:

- o The order of preference of the DOTS signal channel address family and transport protocol (most preferred first) is: UDP over IPv6, UDP over IPv4, TCP over IPv6, and finally TCP over IPv4. This order adheres to the address preference order specified in [RFC6724] and the DOTS signal channel preference which privileges the use of UDP over TCP (to avoid TCP's head of line blocking).
- o The DOTS client after successfully establishing a connection MUST cache information regarding the outcome of each connection attempt for a specific time period, and it uses that information to avoid thrashing the network with subsequent attempts. The cached information is flushed when its age exceeds a specific time period on the order of few minutes (e.g., 10 min). Typically, if the DOTS client has to re-establish the connection with the same DOTS server within few seconds after the Happy Eyeballs mechanism is completed, caching avoids trashing the network especially in the presence of DDoS attack traffic.
- o If DOTS signal channel session is established with TLS (but DTLS failed), the DOTS client periodically repeats the mechanism to discover whether DOTS signal channel messages with DTLS over UDP becomes available from the DOTS server, so the DOTS client can migrate the DOTS signal channel from TCP to UDP. Such probing SHOULD NOT be done more frequently than every 24 hours and MUST NOT be done more frequently than every 5 minutes.

When connection attempts are made during an attack, the DOTS client SHOULD use a "Connection Attempt Delay" [RFC8305] set to 100 ms.

In reference to Figure 4, the DOTS client proceeds with the connection attempts following the rules in [RFC8305]. In this example, it is assumed that the IPv6 path is broken and UDP traffic is dropped by a middlebox but has little impact to the DOTS client because there is no long delay before using IPv4 and TCP.

**Note:**

- * Retransmission messages are not shown.
- * $T1-T0=T2-T1=T3-T2=$ Connection Attempt Delay.

Figure 4: DOTS Happy Eyeballs (Sample Flow)

A single DOTS signal channel between DOTS agents can be used to exchange multiple DOTS signal messages. To reduce DOTS client and DOTS server workload, DOTS clients SHOULD re-use the (D)TLS session.

4.4. DOTS Mitigation Methods

The following methods are used by a DOTS client to request, withdraw, or retrieve the status of mitigation requests:

PUT: DOTS clients use the PUT method to request mitigation from a DOTS server (Section 4.4.1). During active mitigation, DOTS clients may use PUT requests to carry mitigation efficacy updates to the DOTS server (Section 4.4.3).

GET: DOTS clients may use the GET method to subscribe to DOTS server status messages, or to retrieve the list of its mitigations maintained by a DOTS server (Section 4.4.2).

DELETE: DOTS clients use the DELETE method to withdraw a request for mitigation from a DOTS server (Section 4.4.4).

Mitigation request and response messages are marked as Non-confirmable messages (Section 2.2 of [RFC7252]).

DOTS agents MUST NOT send more than one UDP datagram per round-trip time (RTT) to the peer DOTS agent on average following the data transmission guidelines discussed in Section 3.1.3 of [RFC8085].

Requests marked by the DOTS client as Non-confirmable messages are sent at regular intervals until a response is received from the DOTS server. If the DOTS client cannot maintain an RTT estimate, it **MUST** NOT send more than one Non-confirmable request every 3 seconds, and **SHOULD** use an even less aggressive rate whenever possible (case 2 in Section 3.1.3 of [RFC8085]).

JSON encoding of YANG modelled data [RFC7951] is used to illustrate the various methods defined in the following sub-sections. Also, the examples use the Labels defined in Sections 9.6.2, 9.6.3, 9.6.4, and 9.6.5.

4.4.1. Request Mitigation

When a DOTS client requires mitigation for some reason, the DOTS client uses the CoAP PUT method to send a mitigation request to its DOTS server(s) (Figures 5 and 6).

If a DOTS client is entitled to solicit the DOTS service, the DOTS server enables mitigation on behalf of the DOTS client by communicating the DOTS client's request to a mitigator (which may be co-located with the DOTS server) and relaying the feedback of the thus-selected mitigator to the requesting DOTS client.

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

{
  ...
}
```

Figure 5: PUT to Convey DOTS Mitigation Requests

The order of the Uri-Path options is important as it defines the CoAP resource. In particular, 'mid' **MUST** follow 'cuid'.

The additional Uri-Path parameters to those defined in Section 4.2 are as follows:

cuid: Stands for Client Unique Identifier. A globally unique identifier that is meant to prevent collisions among DOTS clients, especially those from the same domain. It **MUST** be generated by DOTS clients.

For the reasons discussed in Appendix A, implementations SHOULD set 'cuid' using the following procedure: first, the Distinguished Encoding Rules (DER)-encoded Abstract Syntax Notation One (ASN.1) representation of the Subject Public Key Info (SPKI) of the DOTS client X.509 certificate [RFC5280], the DOTS client raw public key [RFC7250], the "Pre-Shared Key (PSK) identity" used by the DOTS client in the TLS 1.2 ClientKeyExchange message, or the "identity" used by the DOTS client in the "pre_shared_key" TLS 1.3 extension is input to the SHA-256 [RFC6234] cryptographic hash. Then, the output of the cryptographic hash algorithm is truncated to 16 bytes; truncation is done by stripping off the final 16 bytes. The truncated output is base64url encoded (Section 5 of [RFC4648]) with the trailing "=" removed from the encoding, and the resulting value used as the 'cuid'.

The 'cuid' is intended to be stable when communicating with a given DOTS server, i.e., the 'cuid' used by a DOTS client SHOULD NOT change over time. Distinct 'cuid' values MAY be used by a single DOTS client per DOTS server.

If a DOTS client has to change its 'cuid' for some reason, it MUST NOT do so when mitigations are still active for the old 'cuid'. The 'cuid' SHOULD be 22 characters to avoid DOTS signal message fragmentation over UDP. Furthermore, if that DOTS client created aliases and filtering entries at the DOTS server by means of the DOTS data channel, it MUST delete all the entries bound to the old 'cuid' and re-install them using the new 'cuid'.

DOTS servers MUST return 4.09 (Conflict) error code to a DOTS peer to notify that the 'cuid' is already in-use by another DOTS client. Upon receipt of that error code, a new 'cuid' MUST be generated by the DOTS peer (e.g., using [RFC4122]).

Client-domain DOTS gateways MUST handle 'cuid' collision directly and it is RECOMMENDED that 'cuid' collision is handled directly by server-domain DOTS gateways.

DOTS gateways MAY rewrite the 'cuid' used by peer DOTS clients. Triggers for such rewriting are out of scope.

This is a mandatory Uri-Path parameter.

mid: Identifier for the mitigation request represented with an integer. This identifier MUST be unique for each mitigation request bound to the DOTS client, i.e., the 'mid' parameter value in the mitigation request needs to be unique (per 'cuid')

and DOTS server) relative to the 'mid' parameter values of active mitigation requests conveyed from the DOTS client to the DOTS server.

In order to handle out-of-order delivery of mitigation requests, 'mid' values MUST increase monotonically.

If the 'mid' value has reached 3/4 of ($2^{32} - 1$) (i.e., 3221225471) and no attack is detected, the DOTS client MUST reset 'mid' to 0 to handle 'mid' rollover. If the DOTS client maintains mitigation requests with pre-configured scopes, it MUST re-create them with the 'mid' restarting at 0.

This identifier MUST be generated by the DOTS client.

This is a mandatory Uri-Path parameter.

'cuid' and 'mid' MUST NOT appear in the PUT request message body (Figure 6). The schema in Figure 6 uses the types defined in Section 6. Note that this figure (and other similar figures depicting a schema) are non-normative sketches of the structure of the message.


```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "string"
        ],
        "target-port-range": [
          {
            "lower-port": number,
            "upper-port": number
          }
        ],
        "target-protocol": [
          number
        ],
        "target-fqdn": [
          "string"
        ],
        "target-uri": [
          "string"
        ],
        "alias-name": [
          "string"
        ],
        "lifetime": number,
        "trigger-mitigation": true|false
      }
    ]
  }
}
```

Figure 6: PUT to Convey DOTS Mitigation Requests (Message Body Schema)

The parameters in the CBOR body (Figure 6) of the PUT request are described below:

target-prefix: A list of prefixes identifying resources under attack. Prefixes are represented using Classless Inter-Domain Routing (CIDR) notation [RFC4632]. As a reminder, the prefix length must be less than or equal to 32 (or 128) for IPv4 (or IPv6).

The prefix list **MUST NOT** include broadcast, loopback, or multicast addresses. These addresses are considered as invalid values. In addition, the DOTS server **MUST** validate that target prefixes are

within the scope of the DOTS client domain. Other validation checks may be supported by DOTS servers.

This is an optional attribute.

target-port-range: A list of port numbers bound to resources under attack.

A port range is defined by two bounds, a lower port number (lower-port) and an upper port number (upper-port). When only 'lower-port' is present, it represents a single port number.

For TCP, UDP, Stream Control Transmission Protocol (SCTP) [RFC4960], or Datagram Congestion Control Protocol (DCCP) [RFC4340], a range of ports can be, for example, 0-1023, 1024-65535, or 1024-49151.

This is an optional attribute.

target-protocol: A list of protocols involved in an attack. Values are taken from the IANA protocol registry [proto_numbers].

If 'target-protocol' is not specified, then the request applies to any protocol.

This is an optional attribute.

target-fqdn: A list of Fully Qualified Domain Names (FQDNs) identifying resources under attack [RFC8499].

How a name is passed to an underlying name resolution library is implementation- and deployment-specific. Nevertheless, once the name is resolved into one or multiple IP addresses, DOTS servers MUST apply the same validation checks as those for 'target-prefix'.

The use of FQDNs may be suboptimal because:

- * It induces both an extra load and increased delays on the DOTS server to handle and manage DNS resolution requests.
- * It does not guarantee that the DOTS server will resolve a name to the same IP addresses that the DOTS client does.

This is an optional attribute.

target-uri: A list of Uniform Resource Identifiers (URIs) [RFC3986] identifying resources under attack.

The same validation checks used for 'target-fqdn' MUST be followed by DOTS servers to validate a target URI.

This is an optional attribute.

alias-name: A list of aliases of resources for which the mitigation is requested. Aliases can be created using the DOTS data channel (Section 6.1 of [I-D.ietf-dots-data-channel]), direct configuration, or other means.

An alias is used in subsequent signal channel exchanges to refer more efficiently to the resources under attack.

This is an optional attribute.

lifetime: Lifetime of the mitigation request in seconds. The RECOMMENDED lifetime of a mitigation request is 3600 seconds -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while still making the request expire in a timely manner when the client has unexpectedly quit. DOTS clients MUST include this parameter in their mitigation requests. Upon the expiry of this lifetime, and if the request is not refreshed, the mitigation request is removed. The request can be refreshed by sending the same request again.

A lifetime of '0' in a mitigation request is an invalid value.

A lifetime of negative one (-1) indicates indefinite lifetime for the mitigation request. The DOTS server MAY refuse indefinite lifetime, for policy reasons; the granted lifetime value is returned in the response. DOTS clients MUST be prepared to not be granted mitigations with indefinite lifetimes.

The DOTS server MUST always indicate the actual lifetime in the response and the remaining lifetime in status messages sent to the DOTS client.

This is a mandatory attribute.

trigger-mitigation: If the parameter value is set to 'false', DDoS mitigation will not be triggered for the mitigation request unless the DOTS signal channel session is lost.

If the DOTS client ceases to respond to heartbeat messages, the DOTS server can detect that the DOTS signal channel session is lost. More details are discussed in Section 4.7.

The default value of the parameter is 'true' (that is, the mitigation starts immediately). If 'trigger-mitigation' is not present in a request, this is equivalent to receiving a request with 'trigger-mitigation' set to 'true'.

This is an optional attribute.

In deployments where server-domain DOTS gateways are enabled, identity information about the origin source client domain ('cdid') SHOULD be propagated to the DOTS server. That information is meant to assist the DOTS server to enforce some policies such as grouping DOTS clients that belong to the same DOTS domain, limiting the number of DOTS requests, and identifying the mitigation scope. These policies can be enforced per-client, per-client domain, or both. Also, the identity information may be used for auditing and debugging purposes.

Figure 7 shows an example of a request relayed by a server-domain DOTS gateway.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cdid=7eeaf349529eb55ed50113"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=123"
Content-Format: "application/dots+cbor"

{
  ...
}
```

Figure 7: PUT for DOTS Mitigation Request as Relayed by a DOTS Gateway

A server-domain DOTS gateway SHOULD add the following Uri-Path parameter:

cdid: Stands for Client Domain Identifier. The 'cdid' is conveyed by a server-domain DOTS gateway to propagate the source domain identity from the gateway's client-facing-side to the gateway's server-facing-side, and from the gateway's server-facing-side to the DOTS server. 'cdid' may be used by the final DOTS server for policy enforcement purposes (e.g., enforce a quota on filtering rules). These policies are deployment-specific.

Server-domain DOTS gateways SHOULD support a configuration option to instruct whether 'cdid' parameter is to be inserted.

In order to accommodate deployments that require enforcing per-client policies, per-client domain policies, or a combination thereof, server-domain DOTS gateways instructed to insert the 'cdid' parameter MUST supply the SPKI hash of the DOTS client X.509 certificate, the DOTS client raw public key, or the hash of the "PSK identity" in the 'cdid', following the same rules for generating the hash conveyed in 'cuid', which is then used by the ultimate DOTS server to determine the corresponding client's domain. The 'cdid' generated by a server-domain gateway is likely to be the same as the 'cuid' except if the DOTS message was relayed by a client-domain DOTS gateway or the 'cuid' was generated from a rogue DOTS client.

If a DOTS client is provisioned, for example, with distinct certificates as a function of the peer server-domain DOTS gateway, distinct 'cdid' values may be supplied by a server-domain DOTS gateway. The ultimate DOTS server MUST treat those 'cdid' values as equivalent.

The 'cdid' attribute MUST NOT be generated and included by DOTS clients.

DOTS servers MUST ignore 'cdid' attributes that are directly supplied by source DOTS clients or client-domain DOTS gateways. This implies that first server-domain DOTS gateways MUST strip 'cdid' attributes supplied by DOTS clients. DOTS servers SHOULD support a configuration parameter to identify DOTS gateways that are trusted to supply 'cdid' attributes.

Only single-valued 'cdid' are defined in this document. That is, only the first on-path server-domain DOTS gateway can insert a 'cdid' value. This specification does not allow multiple server-domain DOTS gateways, whenever involved in the path, to insert a 'cdid' value for each server-domain gateway.

This is an optional Uri-Path. When present, 'cdid' MUST be positioned before 'cuid'.

A DOTS gateway MAY add the CoAP Hop-Limit Option [I-D.ietf-core-hop-limit].

Because of the complexity to handle partial failure cases, this specification does not allow for including multiple mitigation requests in the same PUT request. Concretely, a DOTS client MUST NOT

include multiple entries in the 'scope' array of the same PUT request.

FQDN and URI mitigation scopes may be thought of as a form of scope alias, in which the addresses associated with the domain name or URI (as resolved by the DOTS server) represent the scope of the mitigation. Particularly, the IP addresses to which the host subcomponent of authority component of an URI resolves represent the 'target-prefix', the URI scheme represents the 'target-protocol', the port subcomponent of authority component of an URI represents the 'target-port-range'. If the optional port information is not present in the authority component, the default port defined for the URI scheme represents the 'target-port'.

In the PUT request at least one of the attributes 'target-prefix', 'target-fqdn', 'target-uri', or 'alias-name' MUST be present.

Attributes and Uri-Path parameters with empty values MUST NOT be present in a request and render the entire request invalid.

Figure 8 shows a PUT request example to signal that TCP port numbers 80, 8080, and 443 used by 2001:db8:6401::1 and 2001:db8:6401::2 servers are under attack. The presence of 'cdid' indicates that a server-domain DOTS gateway has modified the initial PUT request sent by the DOTS client. Note that 'cdid' MUST NOT appear in the PUT request message body.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cdid=7eeaf349529eb55ed50113"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=123"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          },
          {
            "lower-port": 443
          },
          {
            "lower-port": 8080
          }
        ],
        "target-protocol": [
          6
        ],
        "lifetime": 3600
      }
    ]
  }
}
```

Figure 8: PUT for DOTS Mitigation Request (An Example)

The corresponding CBOR encoding format for the payload is shown in Figure 9.

```

A1          # map(1)
  01        # unsigned(1)
    A1      # map(1)
      02    # unsigned(2)
        81  # array(1)
          A3 # map(3)
            06 # unsigned(6)
              82 # array(2)
                74 # text(20)
                  323030313A6462383A363430313A3A312F313238
                    74 # text(20)
                      323030313A6462383A363430313A3A322F313238
                        07 # unsigned(7)
                          83 # array(3)
                            A1 # map(1)
                              08 # unsigned(8)
                                18 50 # unsigned(80)
                                  A1 # map(1)
                                    08 # unsigned(8)
                                      19 01BB # unsigned(443)
                                        A1 # map(1)
                                          08 # unsigned(8)
                                            19 1F90 # unsigned(8080)
                                              0A # unsigned(10)
                                                81 # array(1)
                                                  06 # unsigned(6)
                                                    0E # unsigned(14)
                                                      19 0E10 # unsigned(3600)

```

Figure 9: PUT for DOTS Mitigation Request (CBOR)

In both DOTS signal and data channel sessions, the DOTS client MUST authenticate itself to the DOTS server (Section 8). The DOTS server MAY use the algorithm presented in Section 7 of [RFC7589] to derive the DOTS client identity or username from the client certificate. The DOTS client identity allows the DOTS server to accept mitigation requests with scopes that the DOTS client is authorized to manage.

The DOTS server couples the DOTS signal and data channel sessions using the DOTS client identity and optionally the 'cdid' parameter value, so the DOTS server can validate whether the aliases conveyed in the mitigation request were indeed created by the same DOTS client using the DOTS data channel session. If the aliases were not created by the DOTS client, the DOTS server MUST return 4.00 (Bad Request) in the response.

The DOTS server couples the DOTS signal channel sessions using the DOTS client identity and optionally the 'cdid' parameter value, and

the DOTS server uses 'mid' and 'cuid' Uri-Path parameter values to detect duplicate mitigation requests. If the mitigation request contains the 'alias-name' and other parameters identifying the target resources (such as 'target-prefix', 'target-port-range', 'target-fqdn', or 'target-uri'), the DOTS server appends the parameter values in 'alias-name' with the corresponding parameter values in 'target-prefix', 'target-port-range', 'target-fqdn', or 'target-uri'.

The DOTS server indicates the result of processing the PUT request using CoAP response codes. CoAP 2.xx codes are success. CoAP 4.xx codes are some sort of invalid requests (client errors). CoAP 5.xx codes are returned if the DOTS server is in an error state or is currently unavailable to provide mitigation in response to the mitigation request from the DOTS client.

Figure 10 shows an example response to a PUT request that is successfully processed by a DOTS server (i.e., CoAP 2.xx response codes). This version of the specification forbids 'cuid' and 'cdid' (if used) to be returned in a response message body.

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

Figure 10: 2.xx Response Body

If the request is missing a mandatory attribute, does not include 'cuid' or 'mid' Uri-Path options, includes multiple 'scope' parameters, or contains invalid or unknown parameters, the DOTS server MUST reply with 4.00 (Bad Request). DOTS agents can safely ignore comprehension-optional parameters they don't understand (Section 9.6.1.1).

A DOTS server that receives a mitigation request with a lifetime set to '0' MUST reply with a 4.00 (Bad Request).

If the DOTS server does not find the 'mid' parameter value conveyed in the PUT request in its configuration data, it MAY accept the mitigation request by sending back a 2.01 (Created) response to the DOTS client; the DOTS server will consequently try to mitigate the attack. A DOTS server could reject mitigation requests when it is

near capacity or needs to rate-limit a particular client, for example.

The relative order of two mitigation requests, having the same 'trigger-mitigation' type, from a DOTS client is determined by comparing their respective 'mid' values. If two mitigation requests with the same 'trigger-mitigation' type have overlapping mitigation scopes, the mitigation request with the highest numeric 'mid' value will override the other mitigation request. Two mitigation requests from a DOTS client have overlapping scopes if there is a common IP address, IP prefix, FQDN, URI, or alias-name. To avoid maintaining a long list of overlapping mitigation requests (i.e., requests with the same 'trigger-mitigation' type and overlapping scopes) from a DOTS client and avoid error-prone provisioning of mitigation requests from a DOTS client, the overlapped lower numeric 'mid' MUST be automatically deleted and no longer available at the DOTS server. For example, if the DOTS server receives a mitigation request which overlaps with an existing mitigation with a higher numeric 'mid', the DOTS server rejects the request by returning 4.09 (Conflict) to the DOTS client. The response includes enough information for a DOTS client to recognize the source of the conflict as described below in the 'conflict-information' subtree with only the relevant nodes listed:

conflict-information: Indicates that a mitigation request is conflicting with another mitigation request. This optional attribute has the following structure:

conflict-cause: Indicates the cause of the conflict. The following values are defined:

- 1: Overlapping targets. 'conflict-scope' provides more details about the conflicting target clauses.

conflict-scope: Characterizes the exact conflict scope. It may include a list of IP addresses, a list of prefixes, a list of port numbers, a list of target protocols, a list of FQDNs, a list of URIs, a list of alias-names, or a 'mid'.

If the DOTS server receives a mitigation request which overlaps with an active mitigation request, but both having distinct 'trigger-mitigation' types, the DOTS server SHOULD deactivate (absent explicit policy/configuration otherwise) the mitigation request with 'trigger-mitigation' set to false. Particularly, if the mitigation request with 'trigger-mitigation' set to false is active, the DOTS server withdraws the mitigation request (i.e., status code is set to '7' as defined in Table 2) and transitions the status of the mitigation request to '8'.

Upon DOTS signal channel session loss with a peer DOTS client, the DOTS server SHOULD withdraw (absent explicit policy/configuration otherwise) any active mitigation requests overlapping with mitigation requests having 'trigger-mitigation' set to false from that DOTS client, as the loss of the session implicitly activates these preconfigured mitigation requests and they take precedence. Note that active-but-terminating period is not observed for mitigations withdrawn at the initiative of the DOTS server.

DOTS clients may adopt various strategies for setting the scopes of immediate and pre-configured mitigation requests to avoid potential conflicts. For example, a DOTS client may tweak pre-configured scopes so that the scope of any overlapping immediate mitigation request will be a subset of the pre-configured scopes. Also, if an immediate mitigation request overlaps with any of the pre-configured scopes, the DOTS client sets the scope of the overlapping immediate mitigation request to be a subset of the pre-configured scopes, so as to get a broad mitigation when the DOTS signal channel collapses and maximize the chance of recovery.

If the request is conflicting with an existing mitigation request from a different DOTS client, the DOTS server may return 2.01 (Created) or 4.09 (Conflict) to the requesting DOTS client. If the DOTS server decides to maintain the new mitigation request, the DOTS server returns 2.01 (Created) to the requesting DOTS client. If the DOTS server decides to reject the new mitigation request, the DOTS server returns 4.09 (Conflict) to the requesting DOTS client. For both 2.01 (Created) and 4.09 (Conflict) responses, the response includes enough information for a DOTS client to recognize the source of the conflict as described below:

conflict-information: Indicates that a mitigation request is conflicting with another mitigation request(s) from other DOTS client(s). This optional attribute has the following structure:

conflict-status: Indicates the status of a conflicting mitigation request. The following values are defined:

- 1: DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently inactive until the conflicts are resolved. Another mitigation request is active.
- 2: DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently active.

- 3: DOTS server has detected conflicting mitigation requests from different DOTS clients. All conflicting mitigation requests are inactive.

conflict-cause: Indicates the cause of the conflict. The following values are defined:

- 1: Overlapping targets. 'conflict-scope' provides more details about the conflicting target clauses.
- 2: Conflicts with an existing accept-list. This code is returned when the DDoS mitigation detects source addresses/prefixes in the accept-listed ACLs are attacking the target.
- 3: CUID Collision. This code is returned when a DOTS client uses a 'cuid' that is already used by another DOTS client. This code is an indication that the request has been rejected and a new request with a new 'cuid' is to be re-sent by the DOTS client (see the example shown in Figure 11). Note that 'conflict-status', 'conflict-scope', and 'retry-timer' MUST NOT be returned in the error response.

conflict-scope: Characterizes the exact conflict scope. It may include a list of IP addresses, a list of prefixes, a list of port numbers, a list of target protocols, a list of FQDNs, a list of URIs, a list of alias-names, or references to conflicting ACLs (by an 'acl-name', typically [I-D.ietf-dots-data-channel]).

retry-timer: Indicates, in seconds, the time after which the DOTS client may re-issue the same request. The DOTS server returns 'retry-timer' only to DOTS client(s) for which a mitigation request is deactivated. Any retransmission of the same mitigation request before the expiry of this timer is likely to be rejected by the DOTS server for the same reasons.

The retry-timer SHOULD be equal to the lifetime of the active mitigation request resulting in the deactivation of the conflicting mitigation request.

If the DOTS server decides to maintain a state for the deactivated mitigation request, the DOTS server updates the lifetime of the deactivated mitigation request to 'retry-timer + 45 seconds' (that is, this mitigation request remains deactivated for the entire duration of 'retry-timer + 45 seconds') so that the DOTS client can refresh the deactivated

mitigation request after 'retry-timer' seconds, but before the expiry of the lifetime, and check if the conflict is resolved.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=7eeaf349529eb55ed50113"
Uri-Path: "mid=12"
```

(1) Request with a conflicting 'cuid'

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "conflict-information": {
          "conflict-cause": "cuid-collision"
        }
      }
    ]
  }
}
```

(2) Message body of the 4.09 (Conflict) response from the DOTS server

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=f30d281ce6b64fc5a0b91e"
Uri-Path: "mid=12"
```

(3) Request with a new 'cuid'

Figure 11: Example of Generating a New 'cuid'

As an active attack evolves, DOTS clients can adjust the scope of requested mitigation as necessary, by refining the scope of resources requiring mitigation. This can be achieved by sending a PUT request with a new 'mid' value that will override the existing one with overlapping mitigation scopes.

For a mitigation request to continue beyond the initial negotiated lifetime, the DOTS client has to refresh the current mitigation request by sending a new PUT request. This PUT request MUST use the same 'mid' value, and MUST repeat all the other parameters as sent in

the original mitigation request apart from a possible change to the lifetime parameter value. In such case, the DOTS server MAY update the mitigation request, and a 2.04 (Changed) response is returned to indicate a successful update of the mitigation request. If this is not the case, the DOTS server MUST reject the request with a 4.00 (Bad Request).

4.4.2. Retrieve Information Related to a Mitigation

A GET request is used by a DOTS client to retrieve information (including status) of DOTS mitigations from a DOTS server.

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

Uri-Path parameters with empty values MUST NOT be present in a request.

The same considerations for manipulating 'cdid' parameter by server-domain DOTS gateways specified in Section 4.4.1 MUST be followed for GET requests.

The 'c' Uri-Query option is used to control selection of configuration and non-configuration data nodes. Concretely, the 'c' (content) parameter and its permitted values defined in the following table [I-D.ietf-core-comi] can be used to retrieve non-configuration data (attack mitigation status), configuration data, or both. The DOTS server MAY support this optional filtering capability. It can safely ignore it if not supported. If the DOTS client supports the optional filtering capability, it SHOULD use "c=n" query (to get back only the dynamically changing data) or "c=c" query (to get back the static configuration values) when the DDoS attack is active to limit the size of the response.

Value	Description
c	Return only configuration descendant data nodes
n	Return only non-configuration descendant data nodes
a	Return all descendant data nodes

The DOTS client can use Block-wise transfer [RFC7959] to get the list of all its mitigations maintained by a DOTS server, it can send Block2 Option in a GET request with NUM = 0 to aid in limiting the size of the response. If the representation of all the active mitigation requests associated with the DOTS client does not fit within a single datagram, the DOTS server MUST use the Block2 Option with NUM = 0 in the GET response. The Size2 Option may be conveyed

in the response to indicate the total size of the resource representation. The DOTS client retrieves the rest of the representation by sending additional GET requests with Block2 Options containing NUM values greater than zero. The DOTS client MUST adhere to the block size preferences indicated by the DOTS server in the response. If the DOTS server uses the Block2 Option in the GET response and the response is for a dynamically changing resource (e.g., "c=n" or "c=a" query), the DOTS server MUST include the ETag Option in the response. The DOTS client MUST include the same ETag value in subsequent GET requests to retrieve the rest of the representation.

The following examples illustrate how a DOTS client retrieves active mitigation requests from a DOTS server. In particular:

- o Figure 12 shows the example of a GET request to retrieve all DOTS mitigation requests signaled by a DOTS client.
- o Figure 13 shows the example of a GET request to retrieve a specific DOTS mitigation request signaled by a DOTS client. The configuration data to be reported in the response is formatted in the same order as was processed by the DOTS server in the original mitigation request.

These two examples assume the default of "c=a"; that is, the DOTS client asks for all data to be reported by the DOTS server.

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

Figure 12: GET to Retrieve all DOTS Mitigation Requests

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

Figure 13: GET to Retrieve a Specific DOTS Mitigation Request

If the DOTS server does not find the 'mid' 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. Likewise, the same error MUST be returned as a response to a request to retrieve all mitigation records (i.e., 'mid' Uri-Path is not defined) of a given DOTS client if the DOTS server does not find any mitigation record for that DOTS client. As a reminder, a DOTS client is identified by its identity (e.g., client certificate, 'cuid') and optionally the 'cdid'.

Figure 14 shows a response example of all active mitigation requests associated with the DOTS client as maintained by the DOTS server. The response indicates the mitigation status of each mitigation request.


```

{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 12332,
        "mitigation-start": "1507818434",
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-protocol": [
          17
        ],
        "lifetime": 1756,
        "status": "attack-successfully-mitigated",
        "bytes-dropped": "134334555",
        "bps-dropped": "43344",
        "pkts-dropped": "333334444",
        "pps-dropped": "432432"
      },
      {
        "mid": 12333,
        "mitigation-start": "1507818393",
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-protocol": [
          6
        ],
        "lifetime": 1755,
        "status": "attack-stopped",
        "bytes-dropped": "0",
        "bps-dropped": "0",
        "pkts-dropped": "0",
        "pps-dropped": "0"
      }
    ]
  }
}

```

Figure 14: Response Body to a GET Request

The mitigation status parameters are described below:

mitigation-start: Mitigation start time is expressed in seconds relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of

[RFC7049]). The CBOR encoding is modified so that the leading tag 1 (epoch-based date/time) MUST be omitted.

This is a mandatory attribute when an attack mitigation is active. Particularly, 'mitigation-start' is not returned for a mitigation with 'status' code set to 8.

lifetime: The remaining lifetime of the mitigation request, in seconds.

This is a mandatory attribute.

status: Status of attack mitigation. The various possible values of 'status' parameter are explained in Table 2.

This is a mandatory attribute.

bytes-dropped: The total dropped byte count for the mitigation request since the attack mitigation is triggered. The count wraps around when it reaches the maximum value of unsigned integer64.

This is an optional attribute.

bps-dropped: The average number of dropped bytes per second for the mitigation request since the attack mitigation is triggered. This average SHOULD be over five-minute intervals (that is, measuring bytes into five-minute buckets and then averaging these buckets over the time since the mitigation was triggered).

This is an optional attribute.

pkts-dropped: The total number of dropped packet count for the mitigation request since the attack mitigation is triggered. The count wraps around when it reaches the maximum value of unsigned integer64.

This is an optional attribute.

pps-dropped: The average number of dropped packets per second for the mitigation request since the attack mitigation is triggered. This average SHOULD be over five-minute intervals (that is, measuring packets into five-minute buckets and then averaging these buckets over the time since the mitigation was triggered).

This is an optional attribute.

Parameter Value	Description
1	Attack mitigation setup is in progress (e.g., changing the network path to redirect the inbound traffic to a DOTS mitigator).
2	Attack is being successfully mitigated (e.g., traffic is redirected to a DDoS mitigator and attack traffic is dropped).
3	Attack has stopped and the DOTS client can withdraw the mitigation request. This status code will be transmitted for immediate mitigation requests till the mitigation is withdrawn or the lifetime expires. For mitigation requests with pre-configured scopes (i.e., 'trigger-mitigation' set to 'false'), this status code will be transmitted 4 times and then transition to "8".
4	Attack has exceeded the mitigation provider capability.
5	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.
6	Attack mitigation is now terminated.
7	Attack mitigation is withdrawn (by the DOTS server). If a mitigation request with 'trigger-mitigation' set to false is withdrawn because it overlaps with an immediate mitigation request, this status code will be transmitted 4 times and then transition to "8" for the mitigation request with pre-configured scopes.
8	Attack mitigation will be triggered for the mitigation request only when the DOTS signal channel session is lost.

Table 2: Values of 'status' Parameter

4.4.2.1. DOTS Servers Sending Mitigation Status

The Observe Option defined in [RFC7641] extends the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server: The client retrieves a representation of the

resource and requests this representation be updated by the server as long as the client is interested in the resource. DOTS implementations MUST use the Observe Option for both 'mitigate' and 'config' (Section 4.2).

A DOTS client conveys the Observe Option set to '0' in the GET request to receive asynchronous notifications of attack mitigation status from the DOTS server.

Unidirectional mitigation notifications within the bidirectional signal channel enables asynchronous notifications between the agents. [RFC7641] indicates that (1) a notification can be sent in a Confirmable or a Non-confirmable message, and (2) the message type used is typically application-dependent and may be determined by the server for each notification individually. For DOTS server application, the message type MUST always be set to Non-confirmable even if the underlying COAP library elects a notification to be sent in a Confirmable message. This overrides the behavior defined in Section 4.5 of [RFC7641] to send a Confirmable message instead of a Non-confirmable message at least every 24 hour for the following reasons: First, the DOTS signal channel uses a heartbeat mechanism to determine if the DOTS client is alive. Second, Confirmable messages are not suitable during an attack.

Due to the higher likelihood of packet loss during a DDoS attack, the DOTS server periodically sends attack mitigation status to the DOTS client and also notifies the DOTS client whenever the status of the attack mitigation changes. If the DOTS server cannot maintain an RTT estimate, it MUST NOT send more than one asynchronous notification every 3 seconds, and SHOULD use an even less aggressive rate whenever possible (case 2 in Section 3.1.3 of [RFC8085]).

When conflicting requests are detected, the DOTS server enforces the corresponding policy (e.g., accept all requests, reject all requests, accept only one request but reject all the others, ...). It is assumed that this policy is supplied by the DOTS server administrator or it is a default behavior of the DOTS server implementation. Then, the DOTS server sends notification message(s) to the DOTS client(s) at the origin of the conflict (refer to the conflict parameters defined in Section 4.4.1). A conflict notification message includes information about the conflict cause, scope, and the status of the mitigation request(s). For example,

- o A notification message with 'status' code set to '7 (Attack mitigation is withdrawn)' and 'conflict-status' set to '1' is sent to a DOTS client to indicate that an active mitigation request is deactivated because a conflict is detected.

- o A notification message with 'status' code set to '1 (Attack mitigation is in progress)' and 'conflict-status' set to '2' is sent to a DOTS client to indicate that this mitigation request is in progress, but a conflict is detected.

Upon receipt of a conflict notification message indicating that a mitigation request is deactivated because of a conflict, a DOTS client MUST NOT resend the same mitigation request before the expiry of 'retry-timer'. It is also recommended that DOTS clients support means to alert administrators about mitigation conflicts.

A DOTS client that is no longer interested in receiving notifications from the DOTS server can simply "forget" the observation. When the DOTS server sends the next notification, the DOTS client will not recognize the token in the message and thus will return a Reset message. This causes the DOTS server to remove the associated entry. Alternatively, the DOTS client can explicitly deregister itself by issuing a GET request that has the Token field set to the token of the observation to be cancelled and includes an Observe Option with the value set to '1' (deregister). The latter is RECOMMENDED.

Figure 15 shows an example of a DOTS client requesting a DOTS server to send notifications related to a mitigation request. Note that for mitigations with pre-configured scopes (i.e., 'trigger-mitigation' set to 'false'), the state will need to transition from 3 (attack-stopped) to 8 (attack-mitigation-signal-loss).

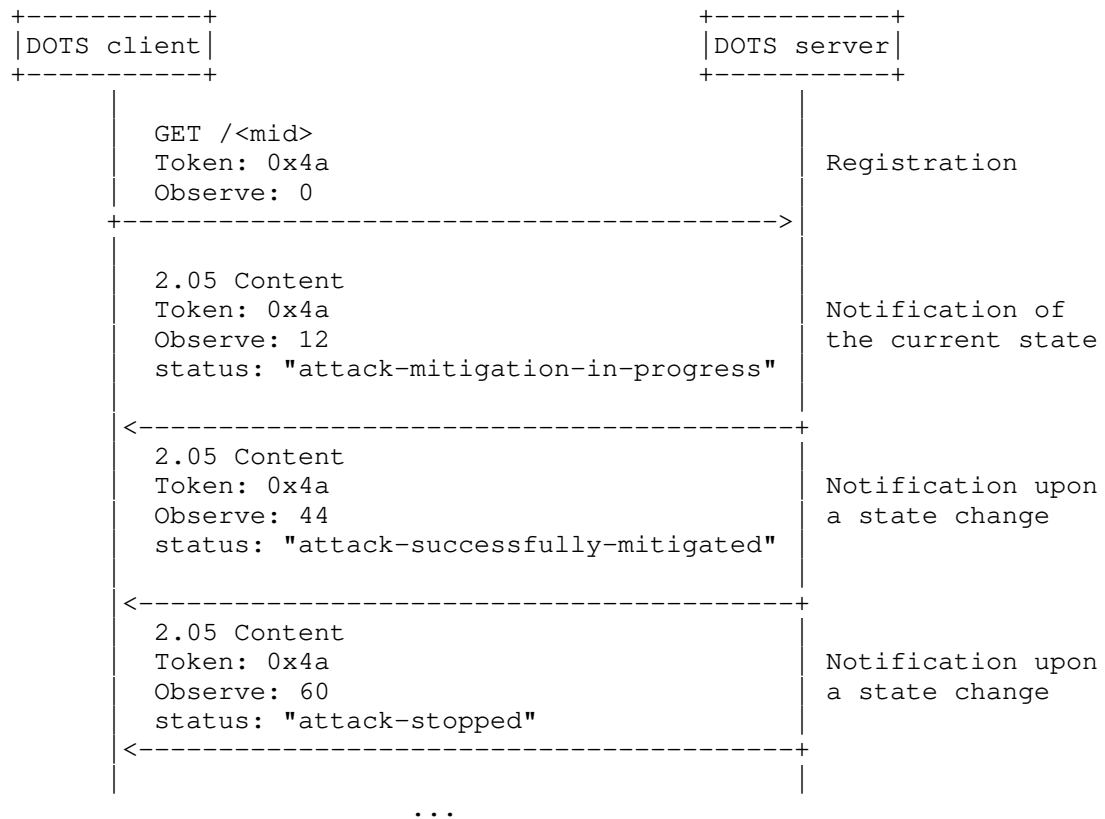


Figure 15: Notifications of Attack Mitigation Status

4.4.2.2. DOTS Clients Polling for Mitigation Status

The DOTS client can send the GET request at frequent intervals without the Observe Option to retrieve the configuration data of the mitigation request and non-configuration data (i.e., the attack status). DOTS clients MAY be configured with a policy indicating the frequency of polling DOTS servers to get the mitigation status. This frequency MUST NOT be more than one UDP datagram per RTT as discussed in Section 3.1.3 of [RFC8085].

If the DOTS server has been able to mitigate the attack and the attack has stopped, the DOTS server indicates as such in the status. In such case, the DOTS client recalls the mitigation request by issuing a DELETE request for this mitigation request (Section 4.4.4).

A DOTS client SHOULD react to the status of the attack as per the information sent by the DOTS server rather than performing its own

detection that the attack has been mitigated. This ensures that the DOTS client does not recall a mitigation request prematurely because it is possible that the DOTS client does not sense the DDoS attack on its resources, but the DOTS server could be actively mitigating the attack because the attack is not completely averted.

4.4.3. Efficacy Update from DOTS Clients

While DDoS mitigation is in progress, due to the likelihood of packet loss, a DOTS client MAY periodically transmit DOTS mitigation efficacy updates to the relevant DOTS server. A PUT request is used to convey the mitigation efficacy update to the DOTS server. This PUT request is treated as a refresh of the current mitigation.

The PUT request used for efficacy update MUST include all the parameters used in the PUT request to carry the DOTS mitigation request (Section 4.4.1) unchanged apart from the 'lifetime' parameter value. If this is not the case, the DOTS server MUST reject the request with a 4.00 (Bad Request).

The If-Match Option (Section 5.10.8.1 of [RFC7252]) with an empty value is used to make the PUT request conditional on the current existence of the mitigation request. If UDP is used as transport, CoAP requests may arrive out-of-order. For example, the DOTS client may send a PUT request to convey an efficacy update to the DOTS server followed by a DELETE request to withdraw the mitigation request, but the DELETE request arrives at the DOTS server before the PUT request. To handle out-of-order delivery of requests, if an If-Match Option is present in the PUT request and the 'mid' in the request matches a mitigation request from that DOTS client, the request is processed by the DOTS server. If no match is found, the PUT request is silently ignored by the DOTS server.

An example of an efficacy update message, which includes an If-Match Option with an empty value, is depicted in Figure 16.

```

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": [
      {
        "target-prefix": [
          "2001:db8:6401::1/128",
          "2001:db8:6401::2/128"
        ],
        "target-port-range": [
          {
            "lower-port": 80
          },
          {
            "lower-port": 443
          },
          {
            "lower-port": 8080
          }
        ],
        "target-protocol": [
          6
        ],
        "attack-status": "under-attack"
      }
    ]
  }
}

```

Figure 16: An Example of Efficacy Update

The 'attack-status' parameter is a mandatory attribute when performing an efficacy update. The various possible values contained in the 'attack-status' parameter are described in Table 3.

Parameter value	Description
1	The DOTS client determines that it is still under attack.
2	The DOTS client determines that the attack is successfully mitigated (e.g., attack traffic is not seen).

Table 3: Values of 'attack-status' Parameter

The DOTS server indicates the result of processing a PUT request using CoAP response codes. The response code 2.04 (Changed) is returned if the DOTS server has accepted the mitigation efficacy update. The error response code 5.03 (Service Unavailable) is returned if the DOTS server has erred or is incapable of performing the mitigation. As specified in [RFC7252], 5.03 uses Max-Age option to indicate the number of seconds after which to retry.

4.4.4. Withdraw a Mitigation

DELETE requests are used to withdraw DOTS mitigation requests from DOTS servers (Figure 17).

'cuid' and 'mid' are mandatory Uri-Path parameters for DELETE requests.

The same considerations for manipulating 'cdid' parameter by DOTS gateways, as specified in Section 4.4.1, MUST be followed for DELETE requests. Uri-Path parameters with empty values MUST NOT be present in a request.

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

Figure 17: Withdraw a DOTS Mitigation

If the DELETE request does not include 'cuid' and 'mid' parameters, the DOTS server MUST reply with a 4.00 (Bad Request).

Once the request is validated, the DOTS server immediately acknowledges a DOTS client's request to withdraw the DOTS signal using 2.02 (Deleted) response code with no response payload. A 2.02 (Deleted) Response Code is returned even if the 'mid' parameter value conveyed in the DELETE request does not exist in its configuration data before the request.

If the DOTS server finds the 'mid' parameter value conveyed in the DELETE request in its configuration data for the DOTS client, then to protect against route or DNS flapping caused by a DOTS client rapidly removing a mitigation, and to dampen the effect of oscillating attacks, the DOTS server MAY allow mitigation to continue for a limited period after acknowledging a DOTS client's withdrawal of a mitigation request. During this period, the DOTS server status messages SHOULD indicate that mitigation is active but terminating (Section 4.4.2).

The initial active-but-terminating period SHOULD be sufficiently long to absorb latency incurred by route propagation. The active-but-terminating period SHOULD be set by default to 120 seconds. If the client requests mitigation again before the initial active-but-terminating period elapses, the DOTS server MAY exponentially increase (the base of the exponent is 2) the active-but-terminating period up to a maximum of 300 seconds (5 minutes).

Once the active-but-terminating period elapses, the DOTS server MUST treat the mitigation as terminated, as the DOTS client is no longer responsible for the mitigation.

If a mitigation is triggered due to a signal channel loss, the DOTS server relies upon normal triggers to stop that mitigation (typically, receipt of a valid DELETE request, expiry of the mitigation lifetime, or scrubbing the traffic to the attack target). In particular, the DOTS server MUST NOT consider the signal channel recovery as a trigger to stop the mitigation.

4.5. DOTS Signal Channel Session Configuration

A DOTS client can negotiate, configure, and retrieve the DOTS signal channel session behavior with its DOTS peers. The DOTS signal channel can be used, for example, to configure the following:

- a. Heartbeat interval (heartbeat-interval): DOTS agents regularly send heartbeats to each other after mutual authentication is successfully completed in order to keep the DOTS signal channel open. Heartbeat messages are exchanged between DOTS agents every 'heartbeat-interval' seconds to detect the current status of the DOTS signal channel session.

- b. Missing heartbeats allowed (missing-hb-allowed): This variable indicates the maximum number of consecutive heartbeat messages for which a DOTS agent did not receive a response before concluding that the session is disconnected or defunct.
- c. Acceptable signal loss ratio: Maximum retransmissions, retransmission timeout value, and other message transmission parameters for the DOTS signal channel.

When the DOTS signal channel is established over a reliable transport (e.g., TCP), there is no need for the reliability mechanisms provided by CoAP over UDP since the underlying TCP connection provides retransmissions and deduplication [RFC8323]. As a reminder, CoAP over reliable transports does not support Confirmable or Non-confirmable message types. As such, the transmission-related parameters (missing-hb-allowed and acceptable signal loss ratio) are negotiated only for DOTS over unreliable transports.

The same or distinct configuration sets may be used during times when a mitigation is active ('mitigating-config') and when no mitigation is active ('idle-config'). This is particularly useful for DOTS servers that might want to reduce heartbeat frequency or cease heartbeat exchanges when an active DOTS client has not requested mitigation. If distinct configurations are used, DOTS agents MUST follow the appropriate configuration set as a function of the mitigation activity (e.g., if no mitigation request is active (also referred to as 'idle' time), 'idle-config'-related values must be followed). Additionally, DOTS agents MUST automatically switch to the other configuration upon a change in the mitigation activity (e.g., if an attack mitigation is launched after an 'idle' time, the DOTS agent switches from 'idle-config' to 'mitigating-config'-related values).

The specification allows for a flexible retry configuration when an unreliable transport is in use. For example, a server may be tweaked to return the following configuration to be used when a mitigation is active:

- o a 'max-retransmit' set to '1' together with a higher 'missing-hb-allowed' value (e.g., 34) and a default 'ack-timeout' set to 2 seconds. This configuration implies more frequent heartbeats in a given time span when a loss is encountered.
- o a lower 'missing-hb-allowed' (e.g., 7) value but delegate the retransmission (with exponential back-off) to the underlying CoAP library by setting 'max-retransmit' to a high value (e.g., 3). When a loss is encountered, this configuration implies less frequent heartbeats compared to the previous bullet.

- o a higher 'ack-timeout' value (e.g., 10 seconds), a 'max-retransmit' set to '1', and a 'missing-hb-allowed' value set to 7. Compared to the previous bullet, this configuration reduces by 50% the number of required heartbeats from the first transmission of a heartbeat message to the time when the DOTS agent gives up.
- o etc.

CoAP Requests and responses are indicated for reliable delivery by marking them as Confirmable messages. DOTS signal channel session configuration requests and responses are marked as Confirmable messages. As explained in Section 2.1 of [RFC7252], a Confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions, until the DOTS server sends an Acknowledgement message (ACK) with the same Message ID conveyed from the DOTS client.

Message transmission parameters are defined in Section 4.8 of [RFC7252]. The DOTS server can either piggyback the response in the acknowledgement message or, if the DOTS server cannot respond immediately to a request carried in a Confirmable message, it simply responds with an Empty Acknowledgement message so that the DOTS client can stop retransmitting the request. Empty Acknowledgement messages are explained in Section 2.2 of [RFC7252]. When the response is ready, the server sends it in a new Confirmable message which in turn needs to be acknowledged by the DOTS client (see Sections 5.2.1 and 5.2.2 of [RFC7252]). Requests and responses exchanged between DOTS agents during 'idle' time are marked as Confirmable messages.

Implementation Note: A DOTS client that receives a response in a Confirmable message may want to clean up the message state right after sending the ACK. If that ACK is lost and the DOTS server retransmits the Confirmable message, the DOTS client may no longer have any state that would help it correlate this response: from the DOTS client's standpoint, the retransmission message is unexpected. The DOTS client will send a Reset message so it does not receive any more retransmissions. This behavior is normal and not an indication of an error (see Section 5.3.2 of [RFC7252] for more details).

4.5.1. Discover Configuration Parameters

A GET request is used to obtain acceptable (e.g., minimum and maximum values) and current configuration parameters on the DOTS server for DOTS signal channel session configuration. This procedure occurs between a DOTS client and its immediate peer DOTS server. As such, this GET request MUST NOT be relayed by a DOTS gateway.

Figure 18 shows how to obtain acceptable configuration parameters for the DOTS server.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "config"
```

Figure 18: GET to Retrieve Configuration

The DOTS server in the 2.05 (Content) response conveys the current, minimum, and maximum attribute values acceptable by the DOTS server (Figure 19).

```
{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "missing-hb-allowed": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "max-retransmit": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      },
      "ack-timeout": {
        "max-value-decimal": "string",
        "min-value-decimal": "string",
        "current-value-decimal": "string"
      },
      "ack-random-factor": {
        "max-value-decimal": "string",
        "min-value-decimal": "string",
        "current-value-decimal": "string"
      }
    },
    "idle-config": {
      "heartbeat-interval": {
        "max-value": number,
        "min-value": number,
        "current-value": number
      }
    }
  }
}
```

```

    },
    "missing-hb-allowed": {
      "max-value": number,
      "min-value": number,
      "current-value": number
    },
    "max-retransmit": {
      "max-value": number,
      "min-value": number,
      "current-value": number
    },
    "ack-timeout": {
      "max-value-decimal": "string",
      "min-value-decimal": "string",
      "current-value-decimal": "string"
    },
    "ack-random-factor": {
      "max-value-decimal": "string",
      "min-value-decimal": "string",
      "current-value-decimal": "string"
    }
  }
}
}

```

Figure 19: GET Configuration Response Body Schema

The parameters in Figure 19 are described below:

mitigating-config: Set of configuration parameters to use when a mitigation is active. The following parameters may be included:

heartbeat-interval: Time interval in seconds between two consecutive heartbeat messages.

'0' is used to disable the heartbeat mechanism.

This is an optional attribute.

missing-hb-allowed: Maximum number of consecutive heartbeat messages for which the DOTS agent did not receive a response before concluding that the session is disconnected.

This is an optional attribute.

max-retransmit: Maximum number of retransmissions for a message (referred to as MAX_RETRANSMIT parameter in CoAP).

This is an optional attribute.

ack-timeout: Timeout value in seconds used to calculate the initial retransmission timeout value (referred to as `ACK_TIMEOUT` parameter in CoAP).

This is an optional attribute.

ack-random-factor: Random factor used to influence the timing of retransmissions (referred to as `ACK_RANDOM_FACTOR` parameter in CoAP).

This is an optional attribute.

idle-config: Set of configuration parameters to use when no mitigation is active. This attribute has the same structure as 'mitigating-config'.

Figure 20 shows an example of acceptable and current configuration parameters on a DOTS server for DOTS signal channel session configuration. The same acceptable configuration is used during mitigation and idle times.

```
{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "max-value": 240,
        "min-value": 15,
        "current-value": 30
      },
      "missing-hb-allowed": {
        "max-value": 9,
        "min-value": 3,
        "current-value": 5
      },
      "max-retransmit": {
        "max-value": 15,
        "min-value": 2,
        "current-value": 3
      },
      "ack-timeout": {
        "max-value-decimal": "30.00",
        "min-value-decimal": "1.00",
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "max-value-decimal": "4.00",
```

```

        "min-value-decimal": "1.10",
        "current-value-decimal": "1.50"
    },
    "idle-config": {
        "heartbeat-interval": {
            "max-value": 240,
            "min-value": 15,
            "current-value": 30
        },
        "missing-hb-allowed": {
            "max-value": 9,
            "min-value": 3,
            "current-value": 5
        },
        "max-retransmit": {
            "max-value": 15,
            "min-value": 2,
            "current-value": 3
        },
        "ack-timeout": {
            "max-value-decimal": "30.00",
            "min-value-decimal": "1.00",
            "current-value-decimal": "2.00"
        },
        "ack-random-factor": {
            "max-value-decimal": "4.00",
            "min-value-decimal": "1.10",
            "current-value-decimal": "1.50"
        }
    }
}

```

Figure 20: Example of a Configuration Response Body

4.5.2. Convey DOTS Signal Channel Session Configuration

A PUT request (Figures 21 and 22) is used to convey the configuration parameters for the signal channel (e.g., heartbeat interval, maximum retransmissions). Message transmission parameters for CoAP are defined in Section 4.8 of [RFC7252]. The RECOMMENDED values of transmission parameter values are ack-timeout (2 seconds), max-retransmit (3), ack-random-factor (1.5). In addition to those parameters, the RECOMMENDED specific DOTS transmission parameter values are 'heartbeat-interval' (30 seconds) and 'missing-hb-allowed' (5).

Note: heartbeat-interval should be tweaked to also assist DOTS messages for NAT traversal (SIG-011 of [RFC8612]). According to [RFC8085], keepalive messages must not be sent more frequently than once every 15 seconds and should use longer intervals when possible. Furthermore, [RFC4787] recommends NATs to use a state timeout of 2 minutes or longer, but experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows. From that standpoint, the RECOMMENDED minimum heartbeat-interval is 15 seconds and the RECOMMENDED maximum heartbeat-interval is 240 seconds. The recommended value of 30 seconds is selected to anticipate the expiry of NAT state.

A heartbeat-interval of 30 seconds may be considered as too chatty in some deployments. For such deployments, DOTS agents may negotiate longer heartbeat-interval values to prevent any network overload with too frequent keepalives.

Different heartbeat intervals can be defined for 'mitigating-config' and 'idle-config' to reduce being too chatty during idle times. If there is an on-path translator between the DOTS client (standalone or part of a DOTS gateway) and the DOTS server, the 'mitigating-config' heartbeat-interval has to be smaller than the translator session timeout. It is recommended that the 'idle-config' heartbeat-interval is also smaller than the translator session timeout to prevent translator traversal issues, or disabled entirely. Means to discover the lifetime assigned by a translator are out of scope.

Section 4.2 of [RFC7252] defines a "CoAP Ping" mechanism. Concretely, the DOTS agent sends an Empty Confirmable message and the peer DOTS agent will respond by sending a Reset message.

When a Confirmable "CoAP Ping" is sent, and if there is no response, the "CoAP Ping" is retransmitted max-retransmit number of times by the CoAP layer using an initial timeout set to a random duration between ack-timeout and (ack-timeout*ack-random-factor) and exponential back-off between retransmissions. By choosing the recommended transmission parameters, the "CoAP Ping" will timeout after 45 seconds. If the DOTS agent does not receive any response from the peer DOTS agent for 'missing-hb-allowed' number of consecutive "CoAP Ping" Confirmable messages, it concludes that the DOTS signal channel session is disconnected. A DOTS client MUST NOT transmit a "CoAP Ping" while waiting for the previous "CoAP Ping" response from the same DOTS server.

If the DOTS agent wishes to change the default values of message transmission parameters, it SHOULD follow the guidance given in

Section 4.8.1 of [RFC7252]. The DOTS agents MUST use the negotiated values for message transmission parameters and default values for non-negotiated message transmission parameters.

The signal channel session configuration is applicable to a single DOTS signal channel session between DOTS agents, so the 'cuid' Uri-Path MUST NOT be used.

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

{
  ...
}
```

Figure 21: PUT to Convey the DOTS Signal Channel Session Configuration Data

The additional Uri-Path parameter to those defined in Table 1 is as follows:

sid: Session Identifier is an identifier for the DOTS signal channel session configuration data represented as an integer. This identifier MUST be generated by DOTS clients. 'sid' values MUST increase monotonically (when a new PUT is generated by a DOTS client to convey the configuration parameters for the signal channel).

This is a mandatory attribute.

```

{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "current-value": number
      },
      "missing-hb-allowed": {
        "current-value": number
      },
      "max-retransmit": {
        "current-value": number
      },
      "ack-timeout": {
        "current-value-decimal": "string"
      },
      "ack-random-factor": {
        "current-value-decimal": "string"
      }
    },
    "idle-config": {
      "heartbeat-interval": {
        "current-value": number
      },
      "missing-hb-allowed": {
        "current-value": number
      },
      "max-retransmit": {
        "current-value": number
      },
      "ack-timeout": {
        "current-value-decimal": "string"
      },
      "ack-random-factor": {
        "current-value-decimal": "string"
      }
    }
  }
}

```

Figure 22: PUT to Convey the DOTS Signal Channel Session Configuration Data (Message Body Schema)

The meaning of the parameters in the CBOR body (Figure 22) is defined in Section 4.5.1.

At least one of the attributes 'heartbeat-interval', 'missing-hb-allowed', 'max-retransmit', 'ack-timeout', and 'ack-random-factor' MUST be present in the PUT request. Note that 'heartbeat-interval',

'missing-hb-allowed', 'max-retransmit', 'ack-timeout', and 'ack-random-factor', if present, do not need to be provided for both 'mitigating-config', and 'idle-config' in a PUT request.

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

Figure 23 shows a PUT request example to convey the configuration parameters for the DOTS signal channel. In this example, the heartbeat mechanism is disabled when no mitigation is active, while the heartbeat interval is set to '91' when a mitigation is active.

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

{
  "ietf-dots-signal-channel:signal-config": {
    "mitigating-config": {
      "heartbeat-interval": {
        "current-value": 91
      },
      "missing-hb-allowed": {
        "current-value": 3
      },
      "max-retransmit": {
        "current-value": 3
      },
      "ack-timeout": {
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "current-value-decimal": "1.50"
      }
    },
    "idle-config": {
      "heartbeat-interval": {
        "current-value": 0
      },
      "max-retransmit": {
        "current-value": 3
      },
      "ack-timeout": {
        "current-value-decimal": "2.00"
      },
      "ack-random-factor": {
        "current-value-decimal": "1.50"
      }
    }
  }
}
```

Figure 23: PUT to Convey the Configuration Parameters

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

- o If the request is missing a mandatory attribute, does not include a 'sid' Uri-Path, or contains one or more invalid or unknown parameters, 4.00 (Bad Request) MUST be returned in the response.
- o If the DOTS server does not find the 'sid' parameter value conveyed in the PUT request in its configuration data and if the DOTS server has accepted the configuration parameters, then a response code 2.01 (Created) MUST be returned in the response.
- o If the DOTS server finds the 'sid' parameter value conveyed in the PUT request in its configuration data and if the DOTS server has accepted the updated configuration parameters, 2.04 (Changed) MUST be returned in the response.
- o If any of the 'heartbeat-interval', 'missing-hb-allowed', 'max-retransmit', 'target-protocol', 'ack-timeout', and 'ack-random-factor' attribute values are not acceptable to the DOTS server, 4.22 (Unprocessable Entity) MUST be returned in the response. Upon receipt of this error code, the DOTS client SHOULD retrieve the maximum and minimum attribute values acceptable to the DOTS server (Section 4.5.1).

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

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

4.5.3. Configuration Freshness and Notifications

Max-Age Option (Section 5.10.5 of [RFC7252]) SHOULD be returned by a DOTS server to associate a validity time with a configuration it sends. This feature allows the update of the configuration data if a change occurs at the DOTS server side. For example, the new configuration may instruct a DOTS client to cease heartbeats or reduce heartbeat frequency.

It is NOT RECOMMENDED to return a Max-Age Option set to 0.

Returning a Max-Age Option set to $2^{32}-1$ is equivalent to associating an infinite lifetime with the configuration.

If a non-zero value of Max-Age Option is received by a DOTS client, it MUST issue a GET request with 'sid' Uri-Path parameter to retrieve the current and acceptable configuration before the expiry of the value enclosed in the Max-Age option. This request is considered by the client and the server as a means to refresh the configuration

parameters for the signal channel. When a DDoS attack is active, refresh requests MUST NOT be sent by DOTS clients and the DOTS server MUST NOT terminate the (D)TLS session after the expiry of the value returned in Max-Age Option.

If Max-Age Option is not returned in a response, the DOTS client initiates GET requests to refresh the configuration parameters each 60 seconds (Section 5.10.5 of [RFC7252]). To prevent such overload, it is RECOMMENDED that DOTS servers return a Max-Age Option in GET responses. Considerations related to which value to use and how such value is set, are implementation- and deployment-specific.

If an Observe Option set to 0 is included in the configuration request, the DOTS server sends notifications of any configuration change (Section 4.2 of [RFC7641]).

If a DOTS server detects that a misbehaving DOTS client does not contact the DOTS server after the expiry of Max-Age and retrieve the signal channel configuration data, it MAY terminate the (D)TLS session. A (D)TLS session is terminated by the receipt of an authenticated message that closes the connection (e.g., a fatal alert (Section 6 of [RFC8446])).

4.5.4. Delete DOTS Signal Channel Session Configuration

A DELETE request is used to delete the installed DOTS signal channel session configuration data (Figure 24).

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

Figure 24: Delete Configuration

The DOTS server resets the DOTS signal channel session configuration back to the default values and acknowledges a DOTS client's request to remove the DOTS signal channel session configuration using 2.02 (Deleted) response code.

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

4.6. Redirected Signaling

Redirected DOTS signaling is discussed in detail in Section 3.2.2 of [I-D.ietf-dots-architecture].

If a DOTS server wants to redirect a DOTS client to an alternative DOTS server for a signal session, then the response code 5.03 (Service Unavailable) will be returned in the response to the DOTS client.

The DOTS server can return the error response code 5.03 in response to a request from the DOTS client or convey the error response code 5.03 in a unidirectional notification response from the DOTS server.

The DOTS server in the error response conveys the alternate DOTS server's FQDN, and the alternate DOTS server's IP address(es) values in the CBOR body (Figure 25).

```
{
  "ietf-dots-signal-channel:redirected-signal": {
    "alt-server": "string",
    "alt-server-record": [
      "string"
    ]
  }
}
```

Figure 25: Redirected Server Error Response Body Schema

The parameters are described below:

alt-server: FQDN of an alternate DOTS server.

This is a mandatory attribute.

alt-server-record: A list of IP addresses of an alternate DOTS server.

This is an optional attribute.

The DOTS server returns the Time to live (TTL) of the alternate DOTS server in a Max-Age Option. That is, the time interval that the alternate DOTS server may be cached for use by a DOTS client. A Max-Age Option set to $2^{32}-1$ is equivalent to receiving an infinite TTL. This value means that the alternate DOTS server is to be used until the alternate DOTS server redirects the traffic with another 5.03 response which encloses an alternate server.

A Max-Age Option set to '0' may be returned for redirecting mitigation requests. Such value means that the redirection applies only for the mitigation request in progress. Returning short TTL in a Max-Age Option may adversely impact DOTS clients on slow links. Returning short values should be avoided under such conditions.

If the alternate DOTS server TTL has expired, the DOTS client MUST use the DOTS server(s), that was provisioned using means discussed in Section 4.1. This fall back mechanism is triggered immediately upon expiry of the TTL, except when a DDoS attack is active.

Requests issued by misbehaving DOTS clients which do not honor the TTL conveyed in the Max-Age Option or react to explicit re-direct messages can be rejected by DOTS servers.

Figure 26 shows a 5.03 response example to convey the DOTS alternate server 'alt-server.example' together with its IP addresses 2001:db8:6401::1 and 2001:db8:6401::2.

```
{
  "ietf-dots-signal-channel:redirected-signal": {
    "alt-server": "alt-server.example",
    "alt-server-record": [
      "2001:db8:6401::1",
      "2001:db8:6401::2"
    ]
  }
}
```

Figure 26: Example of Redirected Server Error Response Body

When the DOTS client receives 5.03 response with an alternate server included, it considers the current request as failed, but SHOULD try re-sending the request to the alternate DOTS server. During a DDoS attack, the DNS server may be the target of another DDoS attack, alternate DOTS server's IP addresses conveyed in the 5.03 response help the DOTS client skip DNS lookup of the alternate DOTS server, at the cost of trusting the first DOTS server to provide accurate information. The DOTS client can then try to establish a UDP or a TCP session with the alternate DOTS server. The DOTS client MAY implement a method to construct IPv4-embedded IPv6 addresses [RFC6052]; this is required to handle the scenario where an IPv6-only DOTS client communicates with an IPv4-only alternate DOTS server.

If the DOTS client has been redirected to a DOTS server to which it has already communicated with within the last five (5) minutes, it MUST ignore the redirection and try to contact other DOTS servers listed in the local configuration or discovered using dynamic means such as DHCP or SRV procedures [I-D.ietf-dots-server-discovery]. It

is RECOMMENDED that DOTS clients support means to alert administrators about redirect loops.

4.7. Heartbeat Mechanism

To provide an indication of signal health and distinguish an 'idle' signal channel from a 'disconnected' or 'defunct' session, the DOTS agent sends a heartbeat over the signal channel to maintain its half of the channel (also, aligned with the "consents" recommendation in Section 6 of [RFC8085]). The DOTS agent similarly expects a heartbeat from its peer DOTS agent, and may consider a session terminated in the prolonged absence of a peer agent heartbeat. Concretely, while the communication between the DOTS agents is otherwise quiescent, the DOTS client will probe the DOTS server to ensure it has maintained cryptographic state and vice versa. Such probes can also keep firewalls and/or stateful translators bindings alive. This probing reduces the frequency of establishing a new handshake when a DOTS signal needs to be conveyed to the DOTS server.

DOTS servers MAY trigger their heartbeat requests immediately after receiving heartbeat probes from peer DOTS clients. As a reminder, it is the responsibility of DOTS clients to ensure that on-path translators/firewalls are maintaining a binding so that the same external IP address and/or port number is retained for the DOTS signal channel session.

In case of a massive DDoS attack that saturates the incoming link(s) to the DOTS client, all traffic from the DOTS server to the DOTS client will likely be dropped, although the DOTS server receives heartbeat requests in addition to DOTS messages sent by the DOTS client. In this scenario, DOTS clients MUST behave differently to handle message transmission and DOTS signal channel session liveliness during link saturation:

The DOTS client MUST NOT consider the DOTS signal channel session terminated even after a maximum 'missing-hb-allowed' threshold is reached. The DOTS client SHOULD keep on using the current DOTS signal channel session to send heartbeat requests over it, so that the DOTS server knows the DOTS client has not disconnected the DOTS signal channel session.

After the maximum 'missing-hb-allowed' threshold is reached, the DOTS client SHOULD try to resume the (D)TLS session. The DOTS client SHOULD send mitigation requests over the current DOTS signal channel session, and in parallel, for example, try to resume the (D)TLS session or use 0-RTT mode in DTLS 1.3 to piggyback the mitigation request in the ClientHello message.

As soon as the link is no longer saturated, if traffic from the DOTS server reaches the DOTS client over the current DOTS signal channel session, the DOTS client can stop (D)TLS session resumption or if (D)TLS session resumption is successful then disconnect the current DOTS signal channel session.

If the DOTS server receives traffic from the peer DOTS client (e.g., peer DOTS client initiated heartbeats) but maximum 'missing-hb-allowed' threshold is reached, the DOTS server MUST NOT consider the DOTS signal channel session disconnected. The DOTS server MUST keep on using the current DOTS signal channel session so that the DOTS client can send mitigation requests over the current DOTS signal channel session. In this case, the DOTS server can identify the DOTS client is under attack and the inbound link to the DOTS client (domain) is saturated. Furthermore, if the DOTS server does not receive a mitigation request from the DOTS client, it implies the DOTS client has not detected the attack or, if an attack mitigation is in progress, it implies the applied DDoS mitigation actions are not yet effective to handle the DDoS attack volume.

If the DOTS server does not receive any traffic from the peer DOTS client, then the DOTS server sends heartbeat requests to the DOTS client and after maximum 'missing-hb-allowed' threshold is reached, the DOTS server concludes the session is disconnected. The DOTS server can then trigger pre-configured mitigation requests for this DOTS client (if any).

In DOTS over UDP, heartbeat messages MUST be exchanged between the DOTS agents using the "CoAP Ping" mechanism defined in Section 4.2 of [RFC7252].

In DOTS over TCP, heartbeat messages MUST be exchanged between the DOTS agents using the Ping and Pong messages specified in Section 5.4 of [RFC8323]. That is, the DOTS agent sends a Ping message and the peer DOTS agent would respond by sending a single Pong message. The sender of a Ping message has to allow up to 'heartbeat-interval' seconds when waiting for a Pong reply. When a failure is detected by a DOTS client, it proceeds with the session recovery following the same approach as the one used for unreliable transports.

5. DOTS Signal Channel YANG Modules

This document defines a YANG [RFC7950] module for DOTS mitigation scope, DOTS signal channel session configuration data, and DOTS redirection signaling.

This YANG module (ietf-dots-signal-channel) defines the DOTS client interaction with the DOTS server as seen by the DOTS client. A DOTS

server is allowed to update the non-configurable 'ro' entities in the responses. This YANG module is not intended to be used via NETCONF/RESTCONF for DOTS server management purposes; such module is out of the scope of this document. It serves only to provide a data model and encoding, but not a management data model.

A companion YANG module is defined to include a collection of types defined by IANA: "iana-dots-signal-channel" (Section 5.2).

5.1. Tree Structure

This document defines the YANG module "ietf-dots-signal-channel" (Section 5.3), which has the following tree structure. A DOTS signal message can be a mitigation, a configuration, or a redirect message.

```

module: ietf-dots-signal-channel
  +--rw dots-signal
    +--rw (message-type)?
      +---:(mitigation-scope)
        +--rw scope* [cuid mid]
          +--rw cdid?          string
          +--rw cuid           string
          +--rw mid            uint32
          +--rw target-prefix* inet:ip-prefix
          +--rw target-port-range* [lower-port]
            +--rw lower-port  inet:port-number
            +--rw upper-port? inet:port-number
          +--rw target-protocol* uint8
          +--rw target-fqdn*    inet:domain-name
          +--rw target-uri*     inet:uri
          +--rw alias-name*     string
          +--rw lifetime?       int32
          +--rw trigger-mitigation? boolean
          +--ro mitigation-start? uint64
          +--ro status?         iana-signal:status
          +--ro conflict-information
            +--ro conflict-status? iana-signal:conflict-status
            +--ro conflict-cause?  iana-signal:conflict-cause
            +--ro retry-timer?     uint32
            +--ro conflict-scope
              +--ro target-prefix*  inet:ip-prefix
              +--ro target-port-range* [lower-port]
                +--ro lower-port  inet:port-number
                +--ro upper-port? inet:port-number
              +--ro target-protocol* uint8
              +--ro target-fqdn*    inet:domain-name
              +--ro target-uri*     inet:uri
              +--ro alias-name*     string

```

```

    +--ro acl-list* [acl-name]
    |   +--ro acl-name
    |   |   -> /ietf-data:dots-data/dots-client/acls/
    |   |       acl/name
    |   +--ro acl-type?
    |   |   -> /ietf-data:dots-data/dots-client/acls/
    |   |       acl/type
    |   +--ro mid?
    |   |   -> ../../../../mid
    +--ro bytes-dropped?
    |   yang:zero-based-counter64
    +--ro bps-dropped?
    |   yang:gauge64
    +--ro pkts-dropped?
    |   yang:zero-based-counter64
    +--ro pps-dropped?
    |   yang:gauge64
    +--rw attack-status?
    |   iana-signal:attack-status
+--:(signal-config)
+--rw sid
|   uint32
+--rw mitigating-config
|   +--rw heartbeat-interval
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16
|   |   +--rw current-value?
|   |   |   uint16
|   +--rw missing-hb-allowed
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16
|   |   +--rw current-value?
|   |   |   uint16
|   +--rw max-retransmit
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16
|   |   +--rw current-value?
|   |   |   uint16
|   +--rw ack-timeout
|   |   +--ro max-value-decimal?
|   |   |   decimal64
|   |   +--ro min-value-decimal?
|   |   |   decimal64
|   |   +--rw current-value-decimal?
|   |   |   decimal64
|   +--rw ack-random-factor
|   |   +--ro max-value-decimal?
|   |   |   decimal64
|   |   +--ro min-value-decimal?
|   |   |   decimal64
|   |   +--rw current-value-decimal?
|   |   |   decimal64
+--rw idle-config
|   +--rw heartbeat-interval
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16
|   |   +--rw current-value?
|   |   |   uint16
|   +--rw missing-hb-allowed
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16
|   |   +--rw current-value?
|   |   |   uint16
|   +--rw max-retransmit
|   |   +--ro max-value?
|   |   |   uint16
|   |   +--ro min-value?
|   |   |   uint16

```

```

|      |  +--rw current-value?   uint16
+--rw  |  ack-timeout
|      |  +--ro max-value-decimal?   decimal64
|      |  +--ro min-value-decimal?   decimal64
|      |  +--rw current-value-decimal? decimal64
+--rw  |  ack-random-factor
|      |  +--ro max-value-decimal?   decimal64
|      |  +--ro min-value-decimal?   decimal64
|      |  +--rw current-value-decimal? decimal64
+--:(redirected-signal)
|      |  +--ro alt-server           string
|      |  +--ro alt-server-record*  inet:ip-address

```

5.2. IANA DOTS Signal Channel YANG Module

```

<CODE BEGINS> file "iana-dots-signal-channel@2019-01-17.yang"
module iana-dots-signal-channel {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-dots-signal-channel";
  prefix iana-signal;

  organization
    "IANA";
  contact
    "Internet Assigned Numbers Authority

    Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA 90094-2536
      United States of America
    Tel:      +1 310 301 5800
    <mailto:iana@iana.org>";
  description
    "This module contains a collection of YANG data types defined
    by IANA and used for DOTS signal channel protocol.

    Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

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

```

```
revision 2019-01-17 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Distributed Denial-of-Service Open Threat
      Signaling (DOTS) Signal Channel Specification";
}

typedef status {
  type enumeration {
    enum attack-mitigation-in-progress {
      value 1;
      description
        "Attack mitigation setup is in progress (e.g., changing
          the network path to re-route the inbound traffic
          to DOTS mitigator).";
    }
    enum attack-successfully-mitigated {
      value 2;
      description
        "Attack is being successfully mitigated (e.g., traffic
          is redirected to a DDoS mitigator and attack
          traffic is dropped or blackholed).";
    }
    enum attack-stopped {
      value 3;
      description
        "Attack has stopped and the DOTS client can
          withdraw the mitigation request.";
    }
    enum attack-exceeded-capability {
      value 4;
      description
        "Attack has exceeded the mitigation provider
          capability.";
    }
    enum dots-client-withdrawn-mitigation {
      value 5;
      description
        "DOTS client has withdrawn the mitigation
          request and the mitigation is active but
          terminating.";
    }
    enum attack-mitigation-terminated {
      value 6;
      description
        "Attack mitigation is now terminated.";
    }
  }
}
```

```
enum attack-mitigation-withdrawn {
    value 7;
    description
        "Attack mitigation is withdrawn.";
}
enum attack-mitigation-signal-loss {
    value 8;
    description
        "Attack mitigation will be triggered
        for the mitigation request only when
        the DOTS signal channel session is lost.";
}
description
    "Enumeration for status reported by the DOTS server.";
}

typedef conflict-status {
    type enumeration {
        enum request-inactive-other-active {
            value 1;
            description
                "DOTS Server has detected conflicting mitigation
                requests from different DOTS clients.
                This mitigation request is currently inactive
                until the conflicts are resolved. Another
                mitigation request is active.";
        }
        enum request-active {
            value 2;
            description
                "DOTS Server has detected conflicting mitigation
                requests from different DOTS clients.
                This mitigation request is currently active.";
        }
        enum all-requests-inactive {
            value 3;
            description
                "DOTS Server has detected conflicting mitigation
                requests from different DOTS clients. All
                conflicting mitigation requests are inactive.";
        }
    }
    description
        "Enumeration for conflict status.";
}

typedef conflict-cause {
```



```
type enumeration {
  enum overlapping-targets {
    value 1;
    description
      "Overlapping targets. conflict-scope provides
       more details about the exact conflict.";
  }
  enum conflict-with-acceptlist {
    value 2;
    description
      "Conflicts with an existing accept-list.

       This code is returned when the DDoS mitigation
       detects that some of the source addresses/prefixes
       listed in the accept-list ACLs are actually
       attacking the target.";
  }
  enum cuid-collision {
    value 3;
    description
      "Conflicts with the cuid used by another
       DOTS client.";
  }
}
description
  "Enumeration for conflict causes.";
}

typedef attack-status {
  type enumeration {
    enum under-attack {
      value 1;
      description
        "The DOTS client determines that it is still under
         attack.";
    }
    enum attack-successfully-mitigated {
      value 2;
      description
        "The DOTS client determines that the attack is
         successfully mitigated.";
    }
  }
  description
    "Enumeration for attack status codes.";
}
}
<CODE ENDS>
```

5.3. IETF DOTS Signal Channel YANG Module

This module uses the common YANG types defined in [RFC6991] and types defined in [I-D.ietf-dots-data-channel].

```
<CODE BEGINS> file "ietf-dots-signal-channel@2019-01-17.yang"
module ietf-dots-signal-channel {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-signal-channel";
  prefix signal;

  import ietf-inet-types {
    prefix inet;
    reference "Section 4 of RFC 6991";
  }
  import ietf-yang-types {
    prefix yang;
    reference "Section 3 of RFC 6991";
  }
  import ietf-dots-data-channel {
    prefix ietf-data;
    reference
      "RFC YYYY: Distributed Denial-of-Service Open Threat Signaling
       (DOTS) Data Channel Specification";
  }
  import iana-dots-signal-channel {
    prefix iana-signal;
  }

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

     Editor:  Konda, Tirumaleswar Reddy
              <mailto:TirumaleswarReddy_Konda@McAfee.com>

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

     Author:  Prashanth Patil
              <mailto:praspati@cisco.com>

     Author:  Andrew Mortensen
              <mailto:amortensen@arbor.net>

     Author:  Nik Teague
```

```
<mailto:nteague@verisign.com>";
description
  "This module contains YANG definition for the signaling
  messages exchanged between a DOTS client and a DOTS server.

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

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

revision 2019-01-17 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Distributed Denial-of-Service Open Threat
    Signaling (DOTS) Signal Channel Specification";
}

/*
 * Groupings
 */

grouping mitigation-scope {
  description
    "Specifies the scope of the mitigation request.";
  list scope {
    key "cuid mid";
    description
      "The scope of the request.";
    leaf cdid {
      type string;
      description
        "The cdid should be included by a server-domain
        DOTS gateway to propagate the client domain
        identification information from the
        gateway's client-facing-side to the gateway's
        server-facing-side, and from the gateway's
        server-facing-side to the DOTS server.

        It may be used by the final DOTS server
```

```
        for policy enforcement purposes.";
    }
    leaf cuid {
        type string;
        description
            "A unique identifier that is
            generated by a DOTS client to prevent
            request collisions. It is expected that the
            cuid will remain consistent throughout the
            lifetime of the DOTS client.";
    }
    leaf mid {
        type uint32;
        description
            "Mitigation request identifier.

            This identifier must be unique for each mitigation
            request bound to the DOTS client.";
    }
    uses ietf-data:target;
    leaf-list alias-name {
        type string;
        description
            "An alias name that points to a resource.";
    }
    leaf lifetime {
        type int32;
        units "seconds";
        default "3600";
        description
            "Indicates the lifetime of the mitigation request.

            A lifetime of '0' in a mitigation request is an
            invalid value.

            A lifetime of negative one (-1) indicates indefinite
            lifetime for the mitigation request.";
    }
    leaf trigger-mitigation {
        type boolean;
        default "true";
        description
            "If set to 'false', DDoS mitigation will not be
            triggered unless the DOTS signal channel
            session is lost.";
    }
    leaf mitigation-start {
        type uint64;
```

```
    config false;
    description
        "Mitigation start time is represented in seconds
        relative to 1970-01-01T00:00:00Z in UTC time.";
}
leaf status {
    type iana-signal:status;
    config false;
    description
        "Indicates the status of a mitigation request.
        It must be included in responses only.";
}
container conflict-information {
    config false;
    description
        "Indicates that a conflict is detected.
        Must only be used for responses.";
    leaf conflict-status {
        type iana-signal:conflict-status;
        description
            "Indicates the conflict status.";
    }
    leaf conflict-cause {
        type iana-signal:conflict-cause;
        description
            "Indicates the cause of the conflict.";
    }
}
leaf retry-timer {
    type uint32;
    units "seconds";
    description
        "The DOTS client must not re-send the
        same request that has a conflict before the expiry of
        this timer.";
}
container conflict-scope {
    description
        "Provides more information about the conflict scope.";
    uses ietf-data:target {
        when "../conflict-cause = 'overlapping-targets'";
    }
    leaf-list alias-name {
        when "../conflict-cause = 'overlapping-targets'";
        type string;
        description
            "Conflicting alias-name.";
    }
    list acl-list {
```

```
when "../../../conflict-cause = 'conflict-with-acceptlist'";
key "acl-name";
description
  "List of conflicting ACLs as defined in the DOTS data
  channel. These ACLs are uniquely defined by
  cuid and acl-name.";
leaf acl-name {
  type leafref {
    path "/ietf-data:dots-data/ietf-data:dots-client/"
      + "ietf-data:acls/ietf-data:acl/ietf-data:name";
  }
  description
    "Reference to the conflicting ACL name bound to
    a DOTS client.";
}
leaf acl-type {
  type leafref {
    path "/ietf-data:dots-data/ietf-data:dots-client/"
      + "ietf-data:acls/ietf-data:acl/ietf-data:type";
  }
  description
    "Reference to the conflicting ACL type bound to
    a DOTS client.";
}
}
leaf mid {
  when "../../../conflict-cause = 'overlapping-targets'";
  type leafref {
    path "../../../mid";
  }
  description
    "Reference to the conflicting 'mid' bound to
    the same DOTS client.";
}
}
leaf bytes-dropped {
  type yang:zero-based-counter64;
  units "bytes";
  config false;
  description
    "The total dropped byte count for the mitigation
    request since the attack mitigation is triggered.
    The count wraps around when it reaches the maximum value
    of counter64 for dropped bytes.";
}
leaf bps-dropped {
  type yang:gauge64;
```

```
    config false;
    description
        "The average number of dropped bits per second for
        the mitigation request since the attack
        mitigation is triggered. This should be over
        five-minute intervals (that is, measuring bytes
        into five-minute buckets and then averaging these
        buckets over the time since the mitigation was
        triggered).";
}
leaf pkts-dropped {
    type yang:zero-based-counter64;
    config false;
    description
        "The total number of dropped packet count for the
        mitigation request since the attack mitigation is
        triggered. The count wraps around when it reaches
        the maximum value of counter64 for dropped packets.";
}
leaf pps-dropped {
    type yang:gauge64;
    config false;
    description
        "The average number of dropped packets per second
        for the mitigation request since the attack
        mitigation is triggered. This should be over
        five-minute intervals (that is, measuring packets
        into five-minute buckets and then averaging these
        buckets over the time since the mitigation was
        triggered).";
}
leaf attack-status {
    type iana-signal:attack-status;
    description
        "Indicates the status of an attack as seen by the
        DOTS client.";
}
}
}

grouping config-parameters {
    description
        "Subset of DOTS signal channel session configuration.";
    container heartbeat-interval {
        description
            "DOTS agents regularly send heartbeats to each other
            after mutual authentication is successfully
            completed in order to keep the DOTS signal channel
```

```
        open.";
    leaf max-value {
        type uint16;
        units "seconds";
        config false;
        description
            "Maximum acceptable heartbeat-interval value.";
    }
    leaf min-value {
        type uint16;
        units "seconds";
        config false;
        description
            "Minimum acceptable heartbeat-interval value.";
    }
    leaf current-value {
        type uint16;
        units "seconds";
        default "30";
        description
            "Current heartbeat-interval value.

            '0' means that heartbeat mechanism is deactivated.";
    }
}
container missing-hb-allowed {
    description
        "Maximum number of missing heartbeats allowed.";
    leaf max-value {
        type uint16;
        config false;
        description
            "Maximum acceptable missing-hb-allowed value.";
    }
    leaf min-value {
        type uint16;
        config false;
        description
            "Minimum acceptable missing-hb-allowed value.";
    }
    leaf current-value {
        type uint16;
        default "5";
        description
            "Current missing-hb-allowed value.";
    }
}
container max-retransmit {
```



```
description
  "Maximum number of retransmissions of a Confirmable
  message.";
leaf max-value {
  type uint16;
  config false;
  description
    "Maximum acceptable max-retransmit value.";
}
leaf min-value {
  type uint16;
  config false;
  description
    "Minimum acceptable max-retransmit value.";
}
leaf current-value {
  type uint16;
  default "3";
  description
    "Current max-retransmit value.";
}
}
container ack-timeout {
  description
    "Initial retransmission timeout value.";
  leaf max-value-decimal {
    type decimal64 {
      fraction-digits 2;
    }
    units "seconds";
    config false;
    description
      "Maximum ack-timeout value.";
  }
  leaf min-value-decimal {
    type decimal64 {
      fraction-digits 2;
    }
    units "seconds";
    config false;
    description
      "Minimum ack-timeout value.";
  }
  leaf current-value-decimal {
    type decimal64 {
      fraction-digits 2;
    }
    units "seconds";
  }
}
```

```
        default "2";
        description
            "Current ack-timeout value.";
    }
}
container ack-random-factor {
    description
        "Random factor used to influence the timing of
        retransmissions.";
    leaf max-value-decimal {
        type decimal64 {
            fraction-digits 2;
        }
        config false;
        description
            "Maximum acceptable ack-random-factor value.";
    }
    leaf min-value-decimal {
        type decimal64 {
            fraction-digits 2;
        }
        config false;
        description
            "Minimum acceptable ack-random-factor value.";
    }
    leaf current-value-decimal {
        type decimal64 {
            fraction-digits 2;
        }
        default "1.5";
        description
            "Current ack-random-factor value.";
    }
}
}

grouping signal-config {
    description
        "DOTS signal channel session configuration.";
    leaf sid {
        type uint32;
        mandatory true;
        description
            "An identifier for the DOTS signal channel
            session configuration data.";
    }
    container mitigating-config {
        description
```

```
        "Configuration parameters to use when a mitigation
        is active.";
    uses config-parameters;
}
container idle-config {
    description
        "Configuration parameters to use when no mitigation
        is active.";
    uses config-parameters;
}
}

grouping redirected-signal {
    description
        "Grouping for the redirected signaling.";
    leaf alt-server {
        type string;
        config false;
        mandatory true;
        description
            "FQDN of an alternate server.";
    }
    leaf-list alt-server-record {
        type inet:ip-address;
        config false;
        description
            "List of records for the alternate server.";
    }
}

/*
 * Main Container for DOTS Signal Channel
 */

container dots-signal {
    description
        "Main container for DOTS signal message.

        A DOTS signal message can be a mitigation, a configuration,
        or a redirected signal message.";
    choice message-type {
        description
            "Can be a mitigation, a configuration, or a redirect
            message.";
        case mitigation-scope {
            description
                "Mitigation scope of a mitigation message.";
            uses mitigation-scope;
        }
    }
}
```

```

    }
    case signal-config {
      description
        "Configuration message.";
      uses signal-config;
    }
    case redirected-signal {
      description
        "Redirected signaling.";
      uses redirected-signal;
    }
  }
}
}
<CODE ENDS>

```

6. YANG/JSON Mapping Parameters to CBOR

All parameters in the payload of the DOTS signal channel MUST be mapped to CBOR types as shown in Table 4 and are assigned an integer key to save space.

- o Note: Implementers must check that the mapping output provided by their YANG-to-CBOR encoding schemes is aligned with the content of Table 4. For example, some CBOR and JSON types for enumerations and the 64-bit quantities can differ depending on the encoder used.

The CBOR key values are divided into two types: comprehension-required and comprehension-optional. DOTS agents can safely ignore comprehension-optional values they don't understand, but cannot successfully process a request if it contains comprehension-required values that are not understood. The 4.00 response SHOULD include a diagnostic payload describing the unknown comprehension-required CBOR key values. The initial set of CBOR key values defined in this specification are of type comprehension-required.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
ietf-dots-signal-channel:mitigation-scope	container	1	5 map	Object
scope	list	2	4 array	Array
cdid	string	3	3 text string	String
cuid	string	4	3 text string	String
mid	uint32	5	0 unsigned	Number

target-prefix	leaf-list	6	4 array	Array
	inet:			
	ip-prefix		3 text string	String
target-port-range	list	7	4 array	Array
lower-port	inet:			
	port-number	8	0 unsigned	Number
upper-port	inet:			
	port-number	9	0 unsigned	Number
target-protocol	leaf-list	10	4 array	Array
	uint8		0 unsigned	Number
target-fqdn	leaf-list	11	4 array	Array
	inet:			
	domain-name		3 text string	String
target-uri	leaf-list	12	4 array	Array
	inet:uri		3 text string	String
alias-name	leaf-list	13	4 array	Array
	string		3 text string	String
lifetime	int32	14	0 unsigned	Number
			1 negative	Number
mitigation-start	uint64	15	0 unsigned	String
status	enumeration	16	0 unsigned	String
conflict-information	container	17	5 map	Object
conflict-status	enumeration	18	0 unsigned	String
conflict-cause	enumeration	19	0 unsigned	String
retry-timer	uint32	20	0 unsigned	Number
conflict-scope	container	21	5 map	Object
acl-list	list	22	4 array	Array
acl-name	leafref	23	3 text string	String
acl-type	leafref	24	3 text string	String
bytes-dropped	yang:zero-based-counter64	25	0 unsigned	String
bps-dropped	yang:gauge64	26	0 unsigned	String
pkts-dropped	yang:zero-based-counter64	27	0 unsigned	String
pps-dropped	yang:gauge64	28	0 unsigned	String
attack-status	enumeration	29	0 unsigned	String
ietf-dots-signal-channel:signal-config	container	30	5 map	Object
sid	uint32	31	0 unsigned	Number
mitigating-config	container	32	5 map	Object
heartbeat-interval	container	33	5 map	Object
max-value	uint16	34	0 unsigned	Number
min-value	uint16	35	0 unsigned	Number
current-value	uint16	36	0 unsigned	Number
missing-hb-allowed	container	37	5 map	Object
max-retransmit	container	38	5 map	Object

ack-timeout	container	39	5 map	Object
ack-random-factor	container	40	5 map	Object
max-value-decimal	decimal64	41	6 tag 4	
			[-2, integer]	String
min-value-decimal	decimal64	42	6 tag 4	
			[-2, integer]	String
current-value-decimal	decimal64	43	6 tag 4	
			[-2, integer]	String
idle-config	container	44	5 map	Object
trigger-mitigation	boolean	45	7 bits 20	False
			7 bits 21	True
ietf-dots-signal-channel:redirected-signal	container	46	5 map	Object
alt-server	string	47	3 text string	String
alt-server-record	leaf-list	48	4 array	Array
	inet:			
	ip-address		3 text string	String

Table 4: CBOR Key Values Used in DOTS Signal Channel Messages & Their Mappings to JSON and YANG

7. (D)TLS Protocol Profile and Performance Considerations

7.1. (D)TLS Protocol Profile

This section defines the (D)TLS protocol profile of DOTS signal channel over (D)TLS and DOTS data channel over TLS.

There are known attacks on (D)TLS, such as man-in-the-middle and protocol downgrade attacks. These are general attacks on (D)TLS and, as such, they are not specific to DOTS over (D)TLS; refer to the (D)TLS RFCs for discussion of these security issues. DOTS agents MUST adhere to the (D)TLS implementation recommendations and security considerations of [RFC7525] except with respect to (D)TLS version. Since DOTS signal channel encryption relying upon (D)TLS is virtually a green-field deployment, DOTS agents MUST implement only (D)TLS 1.2 or later.

When a DOTS client is configured with a domain name of the DOTS server, and connects to its configured DOTS server, the server may present it with a PKIX certificate. In order to ensure proper authentication, a DOTS client MUST verify the entire certification path per [RFC5280]. Additionally, the DOTS client MUST use [RFC6125] validation techniques to compare the domain name with the certificate provided. Certification authorities that issue DOTS server certificates SHOULD support the DNS-ID and SRV-ID identifier types. DOTS server SHOULD prefer the use of DNS-ID and SRV-ID over CN-ID

identifier types in certificate requests (as described in Section 2.3 of [RFC6125]) and the wildcard character '*' SHOULD NOT be included in the presented identifier. DOTS doesn't use URI-IDs for server identity verification.

A key challenge to deploying DOTS is the provisioning of DOTS clients, including the distribution of keying material to DOTS clients to enable the required mutual authentication of DOTS agents. Enrollment over Secure Transport (EST) [RFC7030] defines a method of certificate enrollment by which domains operating DOTS servers may provide DOTS clients with all the necessary cryptographic keying material, including a private key and a certificate to authenticate themselves. One deployment option is DOTS clients behave as EST clients for certificate enrollment from an EST server provisioned by the mitigation provider. This document does not specify which EST or other mechanism the DOTS client uses to achieve initial enrollment.

The Server Name Indication (SNI) extension [RFC6066] defines a mechanism for a client to tell a (D)TLS server the name of the server it wants to contact. This is a useful extension for hosting environments where multiple virtual servers are reachable over a single IP address. The DOTS client may or may not know if it is interacting with a DOTS server in a virtual server hosting environment, so the DOTS client SHOULD include the DOTS server FQDN in the SNI extension.

Implementations compliant with this profile MUST implement all of the following items:

- o DTLS record replay detection (Section 3.3 of [RFC6347]) or an equivalent mechanism to protect against replay attacks.
- o DTLS session resumption without server-side state to resume session and convey the DOTS signal.
- o At least one of raw public keys [RFC7250] or PSK handshake [RFC4279] with (EC)DHE key exchange which reduces the size of the ServerHello, and can be used by DOTS agents that cannot obtain certificates.

Implementations compliant with this profile SHOULD implement all of the following items to reduce the delay required to deliver a DOTS signal channel message:

- o TLS False Start [RFC7918] which reduces round-trips by allowing the TLS client's second flight of messages (ChangeCipherSpec) to also contain the DOTS signal. TLS False Start is formally defined

for use with TLS, but the same technique is applicable to DTLS as well.

- o Cached Information Extension [RFC7924] which avoids transmitting the server's certificate and certificate chain if the client has cached that information from a previous TLS handshake.

Compared to UDP, DOTS signal channel over TCP requires an additional round-trip time (RTT) of latency to establish a TCP connection. DOTS implementations are encouraged to implement TCP Fast Open [RFC7413] to eliminate that RTT.

7.2. (D)TLS 1.3 Considerations

TLS 1.3 provides critical latency improvements for connection establishment over TLS 1.2. The DTLS 1.3 protocol [I-D.ietf-tls-dtls13] is based upon the TLS 1.3 protocol and provides equivalent security guarantees. (D)TLS 1.3 provides two basic handshake modes the DOTS signal channel can take advantage of:

- o A full handshake mode in which a DOTS client can send a DOTS mitigation request message after one round trip and the DOTS server immediately responds with a DOTS mitigation response. This assumes no packet loss is experienced.
- o 0-RTT mode in which the DOTS client can authenticate itself and send DOTS mitigation request messages in the first message, thus reducing handshake latency. 0-RTT only works if the DOTS client has previously communicated with that DOTS server, which is very likely with the DOTS signal channel.

The DOTS client has to establish a (D)TLS session with the DOTS server during 'idle' time and share a PSK.

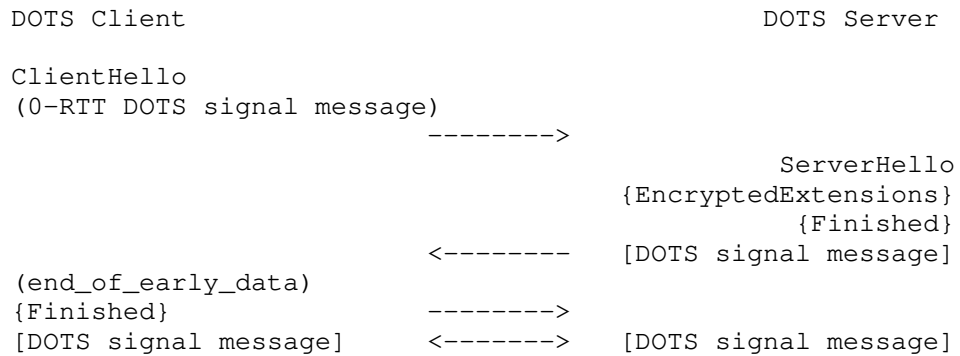
During a DDoS attack, the DOTS client can use the (D)TLS session to convey the DOTS mitigation request message and, if there is no response from the server after multiple retries, the DOTS client can resume the (D)TLS session in 0-RTT mode using PSK.

DOTS servers that support (D)TLS 1.3 MAY allow DOTS clients to send early data (0-RTT). DOTS clients MUST NOT send "CoAP Ping" as early data; such messages MUST be rejected by DOTS servers. Section 8 of [RFC8446] discusses some mechanisms to implement to limit the impact of replay attacks on 0-RTT data. If the DOTS server accepts 0-RTT, it MUST implement one of these mechanisms to prevent replay at the TLS layer. A DOTS server can reject 0-RTT by sending a TLS HelloRetryRequest.

The DOTS signal channel messages sent as early data by the DOTS client are idempotent requests. As a reminder, the Message ID (Section 3 of [RFC7252]) is changed each time a new CoAP request is sent, and the Token (Section 5.3.1 of [RFC7252]) is randomized in each CoAP request. The DOTS server(s) MUST use the Message ID and the Token in the DOTS signal channel message to detect replay of early data at the application layer, and accept 0-RTT data at most once from the same DOTS client. This anti-replay defense requires sharing the Message ID and the Token in the 0-RTT data between DOTS servers in the DOTS server domain. DOTS servers do not rely on transport coordinates to identify DOTS peers. As specified in Section 4.4.1, DOTS servers couple the DOTS signal channel sessions using the DOTS client identity and optionally the 'cdid' parameter value. Furthermore, 'mid' value is monotonically increased by the DOTS client for each mitigation request, attackers replaying mitigation requests with lower numeric 'mid' values and overlapping scopes with mitigation requests having higher numeric 'mid' values will be rejected systematically by the DOTS server. Likewise, 'sid' value is monotonically increased by the DOTS client for each configuration request (Section 4.5.2), attackers replaying configuration requests with lower numeric 'sid' values will be rejected by the DOTS server if it maintains a higher numeric 'sid' value for this DOTS client.

Owing to the aforementioned protections, all DOTS signal channel requests are safe to transmit in TLS 1.3 as early data. Refer to [I-D.boucadair-dots-earlydata] for more details.

A simplified TLS 1.3 handshake with 0-RTT DOTS mitigation request message exchange is shown in Figure 27.



Note that:

- () Indicates messages protected 0-RTT keys
- { } Indicates messages protected using handshake keys
- [] Indicates messages protected using 1-RTT keys

Figure 27: A Simplified TLS 1.3 Handshake with 0-RTT

7.3. DTLS MTU and Fragmentation

To avoid DOTS signal message fragmentation and the subsequent decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record fit within a single datagram. As a reminder, DTLS handles fragmentation and reassembly only for handshake messages and not for the application data (Section 4.1.1 of [RFC6347]). If the PMTU cannot be discovered, DOTS agents MUST assume a PMTU of 1280 bytes, as IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater as specified in [RFC8200]. If IPv4 support on legacy or otherwise unusual networks is a consideration and the PMTU is unknown, DOTS implementations MAY assume on a PMTU of 576 bytes for IPv4 datagrams, as every IPv4 host must be capable of receiving a packet whose length is equal to 576 bytes as discussed in [RFC0791] and [RFC1122].

The DOTS client must consider the amount of record expansion expected by the DTLS processing when calculating the size of CoAP message that fits within the path MTU. Path MTU MUST be greater than or equal to [CoAP message size + DTLS 1.2 overhead of 13 octets + authentication overhead of the negotiated DTLS cipher suite + block padding] (Section 4.1.1.1 of [RFC6347]). If the total request size exceeds the path MTU then the DOTS client MUST split the DOTS signal into separate messages; for example, the list of addresses in the 'target-prefix' parameter could be split into multiple lists and each list conveyed in a new PUT request.

Implementation Note: DOTS choice of message size parameters works well with IPv6 and with most of today's IPv4 paths. However, with IPv4, it is harder to safely make sure that there is no IP fragmentation. If the IPv4 path MTU is unknown, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes, as per [RFC0791].

8. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

(D)TLS based upon client certificate can be used for mutual authentication between DOTS agents. If, for example, a DOTS gateway is involved, DOTS clients and DOTS gateways must perform mutual authentication; only authorized DOTS clients are allowed to send DOTS signals to a DOTS gateway. The DOTS gateway and the DOTS server must perform mutual authentication; a DOTS server only allows DOTS signal channel messages from an authorized DOTS gateway, thereby creating a two-link chain of transitive authentication between the DOTS client and the DOTS server.

The DOTS server should support certificate-based client authentication. The DOTS client should respond to the DOTS server's TLS CertificateRequest message with the PKIX certificate held by the DOTS client. DOTS client certificate validation must be performed as per [RFC5280] and the DOTS client certificate must conform to the [RFC5280] certificate profile. If a DOTS client does not support TLS client certificate authentication, it must support pre-shared key based or raw public key based client authentication.

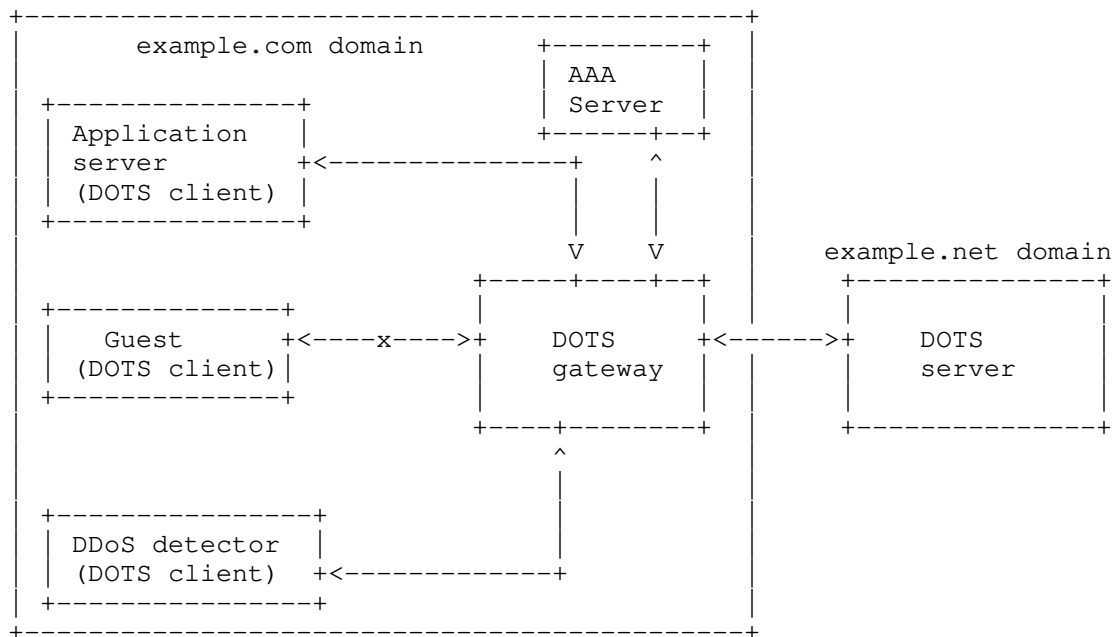


Figure 28: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 28, the DOTS gateway and DOTS clients within the 'example.com' domain mutually authenticate. After the DOTS gateway validates the identity of a DOTS client, it communicates with the AAA server in the 'example.com' domain to determine if the DOTS client is authorized to request DDoS mitigation. If the DOTS client is not authorized, a 4.01 (Unauthorized) is returned in the response to the DOTS client. In this example, the DOTS gateway only allows the application server and DDoS attack detector to request DDoS mitigation, but does not permit the user of type 'guest' to request DDoS mitigation.

Also, DOTS gateways and servers located in different domains must perform mutual authentication (e.g., using certificates). A DOTS server will only allow a DOTS gateway with a certificate for a particular domain to request mitigation for that domain. In reference to Figure 28, the DOTS server only allows the DOTS gateway to request mitigation for 'example.com' domain and not for other domains.

9. IANA Considerations

9.1. DOTS Signal Channel UDP and TCP Port Number

IANA is requested to assign the port number TBD to the DOTS signal channel protocol for both UDP and TCP from the "Service Name and Transport Protocol Port Number Registry" available at <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

The assignment of port number 4646 is strongly suggested, as 4646 is the ASCII decimal value for ".." (DOTS).

9.2. Well-Known 'dots' URI

This document requests IANA to register the 'dots' well-known URI (Table 5) in the Well-Known URIs registry (<https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml>) as defined by [RFC5785]:

URI suffix	Change controller	Specification document(s)	Related information
dots	IETF	[RFCXXXX]	None

Table 5: 'dots' well-known URI

9.3. Media Type Registration

This document requests IANA to register the "application/dots+cbor" media type in the "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838], which can be used to indicate that the content is a DOTS signal channel object:

- o Type name: application
- o Subtype name: dots+cbor
- o Required parameters: N/A
- o Optional parameters: N/A
- o Encoding considerations: binary
- o Security considerations: See the Security Considerations section of [RFCXXXX]
- o Interoperability considerations: N/A
- o Published specification: [RFCXXXX]
- o Applications that use this media type: DOTS agents sending DOTS messages over CoAP over (D)TLS.
- o Fragment identifier considerations: N/A

- o Additional information:
 - Magic number(s): N/A
 - File extension(s): N/A
 - Macintosh file type code(s): N/A
- o Person & email address to contact for further information:
IESG, iesg@ietf.org
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author: See Authors' Addresses section.
- o Change controller: IESG
- o Provisional registration? No

9.4. CoAP Content-Formats Registration

This document requests IANA to register the CoAP Content-Format ID for the "application/dots+cbor" media type in the "CoAP Content-Formats" registry [IANA.CoAP.Content-Formats] (0-255 range):

- o Media Type: application/dots+cbor
- o Encoding: -
- o Id: TBD1
- o Reference: [RFCXXXX]

9.5. CBOR Tag Registration

This section defines the DOTS CBOR tag as another means for applications to declare that a CBOR data structure is a DOTS signal channel object. Its use is optional and is intended for use in cases in which this information would not otherwise be known. DOTS CBOR tag is not required for DOTS signal channel protocol version specified in this document. If present, the DOTS tag MUST prefix a DOTS signal channel object.

This document requests IANA to register the DOTS signal channel CBOR tag in the "CBOR Tags" registry [IANA.CBOR.Tags] using the 24-255 range:

- o CBOR Tag: TBD2 (please assign the same value as the Content-Format)
- o Data Item: DDoS Open Threat Signaling (DOTS) signal channel object
- o Semantics: DDoS Open Threat Signaling (DOTS) signal channel object, as defined in [RFCXXXX]
- o Description of Semantics: [RFCXXXX]

9.6. DOTS Signal Channel Protocol Registry

The document requests IANA to create a new registry, entitled "DOTS Signal Channel Registry". The following sections define sub-registries.

9.6.1. DOTS Signal Channel CBOR Key Values Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel CBOR Key Values".

The structure of this sub-registry is provided in Section 9.6.1.1. Section 9.6.1.2 provides how the registry is initially populated with the values in Table 4.

9.6.1.1. Registration Template

Parameter name:

Parameter name as used in the DOTS signal channel.

CBOR Key Value:

Key value for the parameter. The key value MUST be an integer in the 1-65535 range. The key values of the comprehension-required range (0x0001 - 0x3FFF) and of the comprehension-optional range (0x8000 - 0xBFFF) are assigned by IETF Review (Section 4.8 of [RFC8126]). The key values of the comprehension-optional range (0x4000 - 0x7FFF) are assigned by Specification Required (Section 4.6 of [RFC8126]) and of the comprehension-optional range (0xC000 - 0xFFFF) are reserved for Private Use (Section 4.1 of [RFC8126]).

Registration requests for the 0x4000 - 0x7FFF range are evaluated after a three-week review period on the dots-signal-reg-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published. New registration requests should be sent in the form of an email to the review mailing list; the request should use an appropriate subject (e.g., "Request to register CBOR Key Value for DOTS: example"). IANA will only accept new registrations from the Designated Experts, and will check that review was requested on the mailing list in accordance with these procedures.

Within the review period, the Designated Experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an

explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts includes determining whether the proposed registration duplicates existing functionality, whether it is likely to be of general applicability or whether it is useful only for a single use case, and whether the registration description is clear. IANA must only accept registry updates to the 0x4000 - 0x7FFF range from the Designated Experts and should direct all requests for registration to the review mailing list. It is suggested that multiple Designated Experts be appointed. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

CBOR Major Type:

CBOR Major type and optional tag for the parameter.

Change Controller:

For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., email address) may also be included.

Specification Document(s):

Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

9.6.1.2. Initial Sub-Registry Content

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document (s)
ietf-dots-signal-channel:mitigation-scope	1	5	IESG	[RFCXXXX]
scope	2	4	IESG	[RFCXXXX]
cdid	3	3	IESG	[RFCXXXX]
cuid	4	3	IESG	[RFCXXXX]
mid	5	0	IESG	[RFCXXXX]
target-prefix	6	4	IESG	[RFCXXXX]
target-port-range	7	4	IESG	[RFCXXXX]
lower-port	8	0	IESG	[RFCXXXX]

upper-port	9	0	IESG	[RFCXXXX]
target-protocol	10	4	IESG	[RFCXXXX]
target-fqdn	11	4	IESG	[RFCXXXX]
target-uri	12	4	IESG	[RFCXXXX]
alias-name	13	4	IESG	[RFCXXXX]
lifetime	14	0/1	IESG	[RFCXXXX]
mitigation-start	15	0	IESG	[RFCXXXX]
status	16	0	IESG	[RFCXXXX]
conflict-information	17	5	IESG	[RFCXXXX]
conflict-status	18	0	IESG	[RFCXXXX]
conflict-cause	19	0	IESG	[RFCXXXX]
retry-timer	20	0	IESG	[RFCXXXX]
conflict-scope	21	5	IESG	[RFCXXXX]
acl-list	22	4	IESG	[RFCXXXX]
acl-name	23	3	IESG	[RFCXXXX]
acl-type	24	3	IESG	[RFCXXXX]
bytes-dropped	25	0	IESG	[RFCXXXX]
bps-dropped	26	0	IESG	[RFCXXXX]
pkts-dropped	27	0	IESG	[RFCXXXX]
pps-dropped	28	0	IESG	[RFCXXXX]
attack-status	29	0	IESG	[RFCXXXX]
ietf-dots-signal-channel:signal-config	30	5	IESG	[RFCXXXX]
sid	31	0	IESG	[RFCXXXX]
mitigating-config	32	5	IESG	[RFCXXXX]
heartbeat-interval	33	5	IESG	[RFCXXXX]
min-value	34	0	IESG	[RFCXXXX]
max-value	35	0	IESG	[RFCXXXX]
current-value	36	0	IESG	[RFCXXXX]
missing-hb-allowed	37	5	IESG	[RFCXXXX]
max-retransmit	38	5	IESG	[RFCXXXX]
ack-timeout	39	5	IESG	[RFCXXXX]
ack-random-factor	40	5	IESG	[RFCXXXX]
min-value-decimal	41	6tag4	IESG	[RFCXXXX]
max-value-decimal	42	6tag4	IESG	[RFCXXXX]
current-value-decimal	43	6tag4	IESG	[RFCXXXX]
idle-config	44	5	IESG	[RFCXXXX]
trigger-mitigation	45	7	IESG	[RFCXXXX]
ietf-dots-signal-channel:redirection-signal	46	5	IESG	[RFCXXXX]
alt-server	47	3	IESG	[RFCXXXX]
alt-server-record	48	4	IESG	[RFCXXXX]

Table 6: Initial DOTS Signal Channel CBOR Key Values Registry

9.6.2. Status Codes Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel Status Codes". Codes in this registry are used as valid values of 'status' parameter.

The registry is initially populated with the following values:

Cod e	Label	Description	Reference
1	attack-mitigation-in-progress	Attack mitigation setup is in progress (e.g., changing the network path to redirect the inbound traffic to a DOTS mitigator).	[RFCXXXX]
2	attack-successfully-mitigated	Attack is being successfully mitigated (e.g., traffic is redirected to a DDoS mitigator and attack traffic is dropped).	[RFCXXXX]
3	attack-stopped	Attack has stopped and the DOTS client can withdraw the mitigation request.	[RFCXXXX]
4	attack-exceeded-capability	Attack has exceeded the mitigation provider capability.	[RFCXXXX]

5	dots-client-withdrawn-mitigation	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.	[RFCXXXX]
6	attack-mitigation-terminated	Attack mitigation is now terminated.	[RFCXXXX]
7	attack-mitigation-withdrawn	Attack mitigation is withdrawn.	[RFCXXXX]
8	attack-mitigation-signal-loss	Attack mitigation will be triggered for the mitigation request only when the DOTS signal channel session is lost.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.6.3. Conflict Status Codes Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel Conflict Status Codes". Codes in this registry are used as valid values of 'conflict-status' parameter.

The registry is initially populated with the following values:

Code	Label	Description	Reference
1	request-inactive-other-active	DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently inactive until the conflicts are resolved. Another mitigation request is active.	[RFCXXXX]
2	request-active	DOTS server has detected conflicting mitigation requests from different DOTS clients. This mitigation request is currently active.	[RFCXXXX]
3	all-requests-inactive	DOTS server has detected conflicting mitigation requests from different DOTS clients. All conflicting mitigation requests are inactive.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.6.4. Conflict Cause Codes Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel Conflict Cause Codes". Codes in this registry are used as valid values of 'conflict-cause' parameter.

The registry is initially populated with the following values:

Code	Label	Description	Reference
1	overlapping-targets	Overlapping targets.	[RFCXXXX]
2	conflict-with-acceptlist	Conflicts with an existing accept-list. This code is returned when the DDoS mitigation detects source addresses/prefixes in the accept-listed ACLs are attacking the target.	[RFCXXXX]
3	cuid-collision	CUID Collision. This code is returned when a DOTS client uses a 'cuid' that is already used by another DOTS client.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.6.5. Attack Status Codes Sub-Registry

The document requests IANA to create a new sub-registry, entitled "DOTS Signal Channel Attack Status Codes". Codes in this registry are used as valid values of 'attack-status' parameter.

The registry is initially populated with the following values:

Code	Label	Description	Reference
1	under-attack	The DOTS client determines that it is still under attack.	[RFCXXXX]
2	attack-successfully-mitigated	The DOTS client determines that the attack is successfully mitigated.	[RFCXXXX]

New codes can be assigned via Standards Action [RFC8126].

9.7. DOTS Signal Channel YANG Modules

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-signal-channel
 Registrant Contact: The IESG.
 XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-dots-signal-channel
 Registrant Contact: IANA.
 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 [RFC7950] within the "YANG Parameters" registry.

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

Name: iana-signal
 Namespace: urn:ietf:params:xml:ns:yang:iana-dots-signal-channel
 Maintained by IANA: Y
 Prefix: iana-signal
 Reference: RFC XXXX

This document defines the initial version of the IANA-maintained iana-dots-signal-channel YANG module. IANA is requested to add this note:

Status, conflict status, conflict cause, and attack status values must not be directly added to the iana-dots-signal-channel YANG module. They must instead be respectively added to the "DOTS Status Codes", "DOTS Conflict Status Codes", "DOTS Conflict Cause Codes", and "DOTS Attack Status Codes" registries.

When a 'status', 'conflict-status', 'conflict-cause', or 'attack-status' value is respectively added to the "DOTS Status Codes", "DOTS Conflict Status Codes", "DOTS Conflict Cause Codes", or "DOTS Attack Status Codes" registry, a new "enum" statement must be added to the iana-dots-signal-channel YANG module. The following "enum" statement, and substatements thereof, should be defined:

"enum": Replicates the label from the registry.

"value": Contains the IANA-assigned value corresponding to the 'status', 'conflict-status', 'conflict-cause', or 'attack-status'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry and add the title of the document.

When the iana-dots-signal-channel YANG module is updated, a new "revision" statement must be added in front of the existing revision statements.

IANA is requested to add this note to "DOTS Status Codes", "DOTS Conflict Status Codes", "DOTS Conflict Cause Codes", and "DOTS Attack Status Codes" registries:

When this registry is modified, the YANG module iana-dots-signal-channel must be updated as defined in [RFCXXXX].

10. Security Considerations

High-level DOTS security considerations are documented in [RFC8612] and [I-D.ietf-dots-architecture].

Authenticated encryption MUST be used for data confidentiality and message integrity. The interaction between the DOTS agents requires Datagram Transport Layer Security (DTLS) or Transport Layer Security (TLS) with a cipher suite offering confidentiality protection, and

the guidance given in [RFC7525] MUST be followed to avoid attacks on (D)TLS. The (D)TLS protocol profile used for the DOTS signal channel is specified in Section 7.

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

An attack vector that can be achieved if the 'cuid' is guessable is a misbehaving DOTS client from within the client's domain which uses the 'cuid' of another DOTS client of the domain to delete or alter active mitigations. For this attack vector to happen, the misbehaving client needs to pass the security validation checks by the DOTS server, and eventually the checks of a client-domain DOTS gateway.

A similar attack can be achieved by a compromised DOTS client which can sniff the TLS 1.2 handshake, use the client certificate to identify the 'cuid' used by another DOTS client. This attack is not possible if algorithms such as version 4 Universally Unique Identifiers (UUIDs) in Section 4.4 of [RFC4122] are used to generate the 'cuid' because such UUIDs are not a deterministic function of the client certificate. Likewise, this attack is not possible with TLS 1.3 because most of the TLS handshake is encrypted and the client certificate is not visible to eavesdroppers.

A compromised DOTS client can collude with a DDoS attacker to send mitigation request for a target resource, gets the mitigation efficacy from the DOTS server, and conveys the mitigation efficacy to the DDoS attacker to possibly change the DDoS attack strategy. Obviously, signaling an attack by the compromised DOTS client to the DOTS server will trigger attack mitigation. 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 request mitigation for specific target resources (e.g., an application server is authorized to request mitigation for its IP addresses but a DDoS mitigator can request mitigation for any target resource in the network). Furthermore, DOTS clients are typically co-located on network security services (e.g., firewall) and a compromised security service potentially can do a lot more damage to the network.

Rate-limiting DOTS requests, including those with new 'cuid' values, from the same DOTS client defends against DoS attacks that would result in varying the 'cuid' to exhaust DOTS server resources. Rate-limit policies SHOULD be enforced on DOTS gateways (if deployed) and DOTS servers.

In order to prevent leaking internal information outside a client-domain, DOTS gateways located in the client-domain SHOULD NOT reveal the identification information that pertains to internal DOTS clients (e.g., source IP address, client's hostname) unless explicitly configured to do so.

DOTS servers MUST verify that requesting DOTS clients are entitled to trigger actions on a given IP prefix. That is, only actions on IP resources that belong to the DOTS client's domain MUST be authorized by a DOTS server. The exact mechanism for the DOTS servers to validate that the target prefixes are within the scope of the DOTS client domain is deployment-specific.

The presence of DOTS gateways may lead to infinite forwarding loops, which is undesirable. To prevent and detect such loops, this document uses the Hop-Limit Option.

When FQDNs are used as targets, the DOTS server MUST rely upon DNS privacy enabling protocols (e.g., DNS over TLS [RFC7858] or DoH [RFC8484]) to prevent eavesdroppers from possibly identifying the target resources protected by the DDoS mitigation service, and means to ensure the target FQDN resolution is authentic (e.g., DNSSEC [RFC4034]).

CoAP-specific security considerations are discussed in Section 11 of [RFC7252], while CBOR-related security considerations are discussed in Section 8 of [RFC7049].

11. Contributors

The following individuals have contributed to this document:

- o Jon Shallow, NCC Group, Email: jon.shallow@nccgroup.trust
- o Mike Geller, Cisco Systems, Inc. 3250 Florida 33309 USA, Email: mgeller@cisco.com
- o Robert Moskowitz, HTT Consulting Oak Park, MI 42837 United States, Email: rgm@htt-consult.com
- o Dan Wing, Email: dwing-ietf@fuggles.com

12. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Roman Danyliw, Michael Richardson, Ehud Doron, Kaname Nishizuka, Dave Dolson, Liang Xia, Gilbert Clark, Xialiang Frank, Jim Schaad, Klaus Hartke, Nesredien Suleiman, Stephen Farrell, and Yoshifumi Nishida for the discussion and comments.

The authors would like to give special thanks to Kaname Nishizuka and Jon Shallow for their efforts in implementing the protocol and performing interop testing at IETF Hackathons.

Thanks to the core WG for the recommendations on Hop-Limit and redirect signaling.

Special thanks to Benjamin Kaduk for the detailed AD review.

Thanks to Alexey Melnikov, Adam Roach, Suresh Krishnan, Mirja Kuehlewind, and Alissa Cooper for the review.

13. References

13.1. Normative References

- [I-D.ietf-core-hop-limit] Boucadair, M., K, R., and J. Shallow, "Constrained Application Protocol (CoAP) Hop-Limit Option", draft-ietf-core-hop-limit-04 (work in progress), July 2019.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [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>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <<https://www.rfc-editor.org/info/rfc4279>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<https://www.rfc-editor.org/info/rfc4632>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<https://www.rfc-editor.org/info/rfc5785>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.

- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [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>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [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>>.
- [RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <<https://www.rfc-editor.org/info/rfc7918>>.
- [RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <<https://www.rfc-editor.org/info/rfc7924>>.
- [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>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.
- [RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <<https://www.rfc-editor.org/info/rfc8323>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

13.2. Informative References

- [I-D.boucadair-dots-earlydata]
Boucadair, M. and R. K, "Using Early Data in DOTS", draft-boucadair-dots-earlydata-00 (work in progress), January 2019.

- [I-D.ietf-core-comi]
Veillette, M., Stok, P., Pelov, A., Bierman, A., and I. Petrov, "CoAP Management Interface", draft-ietf-core-comi-07 (work in progress), July 2019.
- [I-D.ietf-core-yang-cbor]
Veillette, M., Petrov, I., and A. Pelov, "CBOR Encoding of Data Modeled with YANG", draft-ietf-core-yang-cbor-10 (work in progress), April 2019.
- [I-D.ietf-dots-architecture]
Mortensen, A., K, R., Andreassen, F., Teague, N., and R. Compton, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-14 (work in progress), May 2019.
- [I-D.ietf-dots-data-channel]
Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-30 (work in progress), July 2019.
- [I-D.ietf-dots-multihoming]
Boucadair, M., K, R., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-02 (work in progress), July 2019.
- [I-D.ietf-dots-server-discovery]
Boucadair, M. and R. K, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Server Discovery", draft-ietf-dots-server-discovery-04 (work in progress), June 2019.
- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-18 (work in progress), July 2019.
- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-32 (work in progress), July 2019.

- [IANA.CBOR.Tags]
IANA, "Concise Binary Object Representation (CBOR) Tags",
<<http://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml>>.
- [IANA.CoAP.Content-Formats]
IANA, "CoAP Content-Formats",
<<http://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>>.
- [IANA.MediaTypees]
IANA, "Media Types",
<<http://www.iana.org/assignments/media-types>>.
- [proto_numbers]
IANA, "Protocol Numbers", 2011,
<<http://www.iana.org/assignments/protocol-numbers>>.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<https://www.rfc-editor.org/info/rfc4340>>.
- [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>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/info/rfc4787>>.

- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<https://www.rfc-editor.org/info/rfc4960>>.
- [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <<https://www.rfc-editor.org/info/rfc4987>>.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <<https://www.rfc-editor.org/info/rfc5389>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <<https://www.rfc-editor.org/info/rfc5925>>.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <<https://www.rfc-editor.org/info/rfc6052>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.

- [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <<https://www.rfc-editor.org/info/rfc6887>>.
- [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <<https://www.rfc-editor.org/info/rfc6888>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/info/rfc7413>>.
- [RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <<https://www.rfc-editor.org/info/rfc7452>>.
- [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<https://www.rfc-editor.org/info/rfc7589>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [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>>.

Appendix A. CUID Generation

The document recommends the use of SPKI to generate the 'cuid'. This design choice is motivated by the following reasons:

- o SPKI is globally unique.
- o It is deterministic.
- o It allows to avoid extra cycles that may be induced by 'cuid' collision.
- o DOTS clients do not need to store the 'cuid' in a persistent storage.
- o It allows to detect compromised DOTS clients that do not adhere to the 'cuid' generation algorithm.

Authors' Addresses

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

Email: kondtir@gmail.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

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

Email: andrew@moretension.com

Nik Teague
Iron Mountain Data Centers
United Kingdom

Email: nteague@ironmountain.co.uk

DOTS
Internet-Draft
Intended status: Standards Track
Expires: November 25, 2019

K. Nishizuka
NTT Communications
M. Boucadair
Orange
T. Reddy
McAfee
T. Nagata
Lepidum
May 24, 2019

Controlling Filtering Rules Using Distributed Denial-of-Service Open
Threat Signaling (DOTS) Signal Channel
draft-ietf-dots-signal-filter-control-01

Abstract

This document specifies an extension to the DOTS signal channel protocol so that DOTS clients can control their filtering rules when an attack mitigation is active.

Particularly, this extension allows a DOTS client to activate or deactivate existing filtering rules during a DDoS attack. The characterization of these filtering rules is supposed to be conveyed by a DOTS client during an idle time by means of the DOTS data channel protocol.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

- o "This version of this YANG module is part of RFC XXXX;"
- o "RFC XXXX: Controlling Filtering Rules Using Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel";
- o reference: RFC XXXX
- o [RFCXXXX]

Please update these statements with the RFC number to be assigned to the following documents:

- o "RFC SSSS: Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification" (used to be [I-D.ietf-dots-signal-channel])

- o "RFC DDDD: Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification" (used to be [I-D.ietf-dots-data-channel])

Please update the "revision" date of the YANG module.

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 25, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. The Problem	3
1.2. The Solution	4
2. Terminology	5
3. Controlling Filtering Rules of a DOTS Client	5
3.1. Binding DOTS Data and Signal Channels	5
3.2. DOTS Signal Channel Extension	6
3.2.1. Parameters and Behaviors	6

3.2.2. DOTS Signal Filtering Control Module	9
3.2.2.1. Tree Structure	9
3.2.2.2. YANG Module	10
4. Sample Examples	12
4.1. Conflict Handling	12
4.2. On-Demand Activation of an Accept-List Filter	16
4.3. DOTS Servers/Mitigators Lacking Capacity	18
5. IANA Considerations	22
5.1. DOTS Signal Channel CBOR Mappings Registry	22
5.2. DOTS Signal Filtering Control YANG Module	23
6. Security Considerations	23
7. Acknowledgements	24
8. References	24
8.1. Normative References	24
8.2. Informative References	24
Authors' Addresses	25

1. Introduction

1.1. The Problem

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

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

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

[I-D.ietf-dots-signal-channel] is designed so that the DDoS server notifies the conflict to the DOTS client (that is, 'conflict-cause' parameter set to 2 (Conflicts with an existing accept list)), but the DOTS client may not be able to withdraw the accept-list rules during the attack period due to the high-volume attack traffic saturating the inbound link to the DOTS client domain. In other words, the DOTS client cannot use the DOTS data channel protocol to withdraw the accept-list filters when a DDoS attack is in progress. This assumes that this DOTS client is the owner of the filtering rule.

1.2. The Solution

This specification addresses the problems discussed in Section 1.1 by adding the capability of managing filtering rules using the DOTS signal channel protocol, which enables a DOTS client to request the activation (or deactivation) of filtering rules during a DDoS attack.

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

Note: The experiment at the IETF103 hackathon [Interop] showed that even when the inbound link is saturated by DDoS attack traffic, the DOTS client can signal mitigation requests using the DOTS signal channel over the saturated link.

Conflicts that are induced by filters installed by other DOTS clients of the same domain are not discussed in this specification.

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

Sample examples are provided in Section 4, in particular:

- o Section 4.1 illustrates how the filter control extension is used when conflicts with Access Control List (ACLs) are detected and reported by a DOTS server.
- o Section 4.2 shows how a DOTS client can instruct a DOTS server to safely forward some specific traffic in 'attack' time.
- o Section 4.3 shows how a DOTS client can react if the DDoS traffic is still being forwarded to the DOTS client domain even if mitigation requests were sent to a DOTS server.

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

2. Terminology

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

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

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

3. Controlling Filtering Rules of a DOTS Client

3.1. Binding DOTS Data and Signal Channels

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

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

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

A filtering rule controlled by the DOTS signal channel is identified by its ACL name (Section 7.2 of [I-D.ietf-dots-data-channel]). Note that an ACL name unambiguously identifies an ACL bound to a DOTS client, but the same name may be used by distinct DOTS clients.

The activation or deactivation of an ACL by the DOTS signal channel overrides the 'activation-type' (defined in Section 7.2 of

[I-D.ietf-dots-data-channel]) a priori conveyed with the filtering rules using the DOTS data channel protocol.

3.2. DOTS Signal Channel Extension

3.2.1. Parameters and Behaviors

This specification extends the mitigation request defined in Section 4.4.1 of [I-D.ietf-dots-signal-channel] to convey the intended control of configured filtering rules. Concretely, the DOTS client conveys 'acl-list' attribute with the following sub-attributes in the CBOR body of a mitigation request (see the YANG-encoded structure in Section 3.2.2.1):

acl-name: A name of an access list defined using the DOTS data channel (Section 7.2 of [I-D.ietf-dots-data-channel]) that is associated with the DOTS client.

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

This is an optional attribute.

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

As a reminder, this attribute can be set to 'deactivate', 'immediate', or 'activate-when-mitigating' as defined in [I-D.ietf-dots-data-channel].

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

This is an optional attribute.

The JSON/YANG mapping to CBOR for 'activation-type' is shown in Table 1.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
activation-type	enumeration	0x0031 (TBD1)	0 unsigned	String

Table 1: JSON/YANG mapping to CBOR for 'activation-type'

By default, ACL-related operations are achieved using the DOTS data channel protocol when no attack is ongoing. DOTS clients MUST NOT use the filtering control over DOTS signal channel in 'idle' time; such requests MUST be discarded by DOTS servers with 4.00 (Bad Request).

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

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

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

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

As a reminder, the 'acl-list' and 'acl-name' parameters are defined as comprehension-required parameters in Table 6 of [I-D.ietf-dots-signal-channel]. Also, the 'activation-type' is defined as a comprehension-required parameter (Section 5.1). Following the rules in Section 6 of [I-D.ietf-dots-signal-channel],

if the DOTS server does not understand the 'acl-list' or 'acl-name' or 'activation-type' attributes, it responds with a "4.00 (Bad Request)" error response code.

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

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

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

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

If the DOTS client receives a 5.03 (Service Unavailable) with a diagnostic payload indicating a failed ACL update as a response to an initial mitigation or a mitigation with adjusted scope, the DOTS client MUST immediately send a new request which repeats all the parameters as sent in the failed mitigation request but without

including the ACL attributes. After the expiry of Max-Age returned in the 5.03 (Service Unavailable) response, the DOTS client retries with a new mitigation request (i.e., a new 'mid') that repeats all the parameters as sent in the failed mitigation request.

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

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

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

3.2.2. DOTS Signal Filtering Control Module

3.2.2.1. Tree Structure

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

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

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

3.2.2.2. YANG Module

This module uses types defined in [I-D.ietf-dots-data-channel].

```
<CODE BEGINS> file "ietf-dots-signal-control@2019-05-13.yang"

module ietf-dots-signal-control {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-dots-signal-control";
  prefix signal-control;

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

  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:   Konda, Tirumaleswar Reddy
              <mailto:TirumaleswarReddy_Konda@McAfee.com>

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

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

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

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

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

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

```
revision 2019-05-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Controlling Filtering Rules Using Distributed
      Denial-of-Service Open Threat Signaling (DOTS)
      Signal Channel";
}

feature control-filtering {
  description
    "This feature means that the DOTS signal channel is able
      to manage the filtering rules created by the same DOTS
      client using the DOTS data channel.";
}

augment "/ietf-signal:dots-signal/ietf-signal:message-type"
  + "/ietf-signal:mitigation-scope/ietf-signal:scope" {
  if-feature control-filtering;

  description "ACL name and activation type.";

  list acl-list {
    key "acl-name";
    description
      "List of ACLs as defined using the DOTS data
        channel. ACLs bound to a DOTS client are uniquely
        identified by a name.";
    leaf acl-name {
      type leafref {
        path "/ietf-data:dots-data/ietf-data:dots-client"
          + "/ietf-data:acls/ietf-data:acl/ietf-data:name";
      }
    }
    description
      "Reference to the ACL name bound to a DOTS client.";
```

```
    }  
    leaf activation-type {  
      type ietf-data:activation-type;  
      default "activate-when-mitigating";  
      description  
        "Sets the activation type of an ACL.";  
    }  
  }  
}  
}  
<CODE ENDS>
```

4. Sample Examples

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

4.1. Conflict Handling

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

```

PUT /restconf/data/ietf-dots-data-channel:dots-data\
  /dots-client=paL8p4Zqo4SLv64TLPXrxA/acls\
  /acl=an-accept-list HTTP/1.1
Host: {host}:{port}
Content-Type: application/yang-data+json

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

```

Figure 2: DOTS Data Channel Request to Create a Filtering

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


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

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

Figure 3: DOTS Signal Channel Mitigation Request

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

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

Figure 4: Status Response (Message Body)

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

indicate that this mitigation request is in progress, but a conflict is detected.

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

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

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

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

Figure 5: PUT for Deactivating a Conflicting Filter

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

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

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

Figure 6: DOTS Data Channel GET Response after Mitigation

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

Let's consider a DOTS client which contacts its DOTS server during 'idle' time to install an accept-list allowing for UDP traffic issued from 2001:db8:1234::/48 to be forwarded to 2001:db8:6401::2/127. It

does so by sending, for example, a PUT request shown in Figure 7. The DOTS server installs this filter with a "deactivated" state.

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\  
  /dots-client=ioiuLoZqo4SLv64TLPXrxA/acls\  
  /acl=my-accept-list HTTP/1.1  
Host: {host}:{port}  
Content-Type: application/yang-data+json  
  
{  
  "ietf-dots-data-channel:acls": {  
    "acl": [  
      {  
        "name": "my-accept-list",  
        "type": "ipv6-acl-type",  
        "activation-type": "deactivate",  
        "aces": {  
          "ace": [  
            {  
              "name": "an-ace",  
              "matches": {  
                "ipv6": {  
                  "destination-ipv6-network": "2001:db8:6401::2/127",  
                  "source-ipv6-network": "2001:db8:1234::/48",  
                  "protocol": 17  
                }  
              },  
              "actions": {  
                "forwarding": "accept"  
              }  
            }  
          ]  
        }  
      ]  
    }  
  }  
}
```

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

Sometime later, consider that a UDP DDoS attack is detected by the DOTS client on 2001:db8:6401::2/127 but the DOTS client wants to let the traffic from 2001:db8:1234::/48 to be accept-listed to the DOTS client domain. Consequently, the DOTS client sends a mitigation request to its DOTS server as shown in Figure 8.

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

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

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

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

4.3. DOTS Servers/Mitigators Lacking Capacity

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

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

```
PUT /restconf/data/ietf-dots-data-channel:dots-data\  
  /dots-client=OopPisZqo4SLv64TLPXrxA/acls\  
  /acl=my-ratelimit-list HTTP/1.1  
Host: {host}:{port}  
Content-Type: application/yang-data+json  
  
{  
  "ietf-dots-data-channel:acls": {  
    "acl": [  
      {  
        "name": "my-ratelimit-list",  
        "type": "ipv6-acl-type",  
        "activation-type": "deactivate",  
        "aces": {  
          "ace": [  
            {  
              "name": "my-ace",  
              "matches": {  
                "ipv6": {  
                  "destination-ipv6-network": "2001:db8:123::/48"  
                }  
              },  
              "actions": {  
                "forwarding": "accept",  
                "rate-limit": "20.00"  
              }  
            }  
          ]  
        }  
      ]  
    }  
  }  
}
```

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

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

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

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

Figure 10: DOTS Signal Channel Mitigation Request to Activate a Rate-Limit Filter

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

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

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

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

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

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


```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=OopPisZqo4SLv64TLPXrxA"
Uri-Path: "mid=87"
Content-Format: "application/dots+cbor"

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

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

5. IANA Considerations

5.1. DOTS Signal Channel CBOR Mappings Registry

This specification registers the 'activation-type' parameter in the IANA "DOTS Signal Channel CBOR Key Values" registry established by [I-D.ietf-dots-signal-channel].

- o Note to the RFC Editor: Please delete (TBD1) once the CBOR key is assigned from the (0x0001 - 0x3FFF) range.

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document (s)
activation-type	0x0031 (TBD1)	0	IESG	[RFCXXXX]

5.2. DOTS Signal Filtering Control YANG Module

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-dots-signal-control

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [RFC7950].

Name: ietf-dots-signal-control

Namespace: urn:ietf:params:xml:ns:yang:ietf-dots-signal-control

Maintained by IANA: N

Prefix: signal-control

Reference: RFC XXXX

6. Security Considerations

The security considerations discussed in [I-D.ietf-dots-signal-channel] and [I-D.ietf-dots-data-channel] need to be taken into account.

A DOTS client is entitled to access only to resources it creates. In particular, a DOTS client can not tweak filtering rules created by other DOTS clients of the same DOTS client domain.

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

7. Acknowledgements

Many thanks to Takahiko Nagata, Wei Pan, Xia Liang, Jon Shallow, and Dan Wing for the comments.

8. References

8.1. Normative References

- [I-D.ietf-dots-data-channel]
Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-29 (work in progress), May 2019.
- [I-D.ietf-dots-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-34 (work in progress), May 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [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.

- [Interop] Nishizuka, K., Shallow, J., and L. Xia , "DOTS Interop test report, IETF 103 Hackathon", November 2018, <<https://datatracker.ietf.org/meeting/103/materials/slides-103-dots-interop-report-from-ietf-103-hackathon-00>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Authors' Addresses

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

Email: kaname@nttv6.jp

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

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

Email: kondtir@gmail.com

Takahiko Nagata
Lepidum
Japan

Email: nagata@lepidum.co.jp

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

T. Reddy
McAfee
M. Boucadair
Orange
E. Doron
Radware Ltd.
July 5, 2019

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry
draft-reddy-dots-telemetry-00

Abstract

This document aims to enrich DOTS signal channel protocol with various telemetry attributes allowing optimal DDoS attack mitigation. This document specifies the normal traffic baseline and attack traffic telemetry attributes a DOTS client can convey to its DOTS server in the mitigation request, the mitigation status telemetry attributes a DOTS server can communicate to a DOTS client, and the mitigation efficacy telemetry attributes a DOTS client can communicate to a DOTS server. The telemetry attributes can assist the mitigator to choose the DDoS mitigation techniques and perform optimal DDoS attack mitigation.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 6, 2020.

Copyright Notice

Copyright (c) 2019 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	4
3. DOTS Telemetry: Overview & Purpose	5
4. DOTS Telemetry Attributes	8
4.1. Pre-mitigation DOTS Telemetry Attributes	8
4.1.1. Total Traffic Normal Baseline	8
4.1.2. Total Pipe Capability	9
4.1.3. Total Attack Traffic	9
4.1.4. Total Traffic	9
4.1.5. Attack Details	9
4.2. DOTS Client to Server Mitigation Efficacy DOTS Telemetry Attributes	10
4.2.1. Total Attack Traffic	10
4.2.2. Attack Details	10
4.3. DOTS Server to Client Mitigation Status DOTS Telemetry Attributes	10
4.3.1. Mitigation Status	10
5. DOTS Telemetry YANG Module	10
5.1. Tree Structure	10
5.2. YANG Module	11
6. IANA Considerations	11
6.1. DOTS Signal Channel CBOR Mappings Registry	11
6.2. DOTS Signal Telemetry YANG Module	11
7. Security Considerations	12
8. Acknowledgements	12
9. References	12
9.1. Normative References	12
9.2. Informative References	13
Authors' Addresses	13

1. Introduction

The Internet security 'battle' between the adversary and security countermeasures is an everlasting one. DDoS attacks have become more vicious and sophisticated in almost all aspects of their maneuvers and malevolent intentions. IT organizations and service providers

are facing DDoS attacks that fall into two broad categories: Network/Transport layer attacks and Application layer attacks. Network/Transport layer attacks target the victim's infrastructure. These attacks are not necessarily aimed at taking down the actual delivered services, but rather to eliminate various network elements (routers, switches, firewalls, transit links, and so on) from serving legitimate user traffic. The main method of such attacks is to send a large volume or high PPS of traffic toward the victim's infrastructure. Typically, attack volumes may vary from a few 100 Mbps/PPS to 100s of Gbps or even Tbps. Attacks are commonly carried out leveraging botnets and attack reflectors for amplification attacks, such as NTP, DNS, SNMP, SSDP, and so on. Application layer attacks target various applications. Typical examples include attacks against HTTP/HTTPS, DNS, SIP, SMTP, and so on. However, all valid applications with their port numbers open at network edges can be attractive attack targets. Application layer attacks are considered more complex and hard to categorize, therefore harder to detect and mitigate efficiently.

To compound the problem, attackers also leverage multi-vector attacks. These merciless attacks are assembled from dynamic attack vectors (Network/Application) and tactics. As such, multiple attack vectors formed by multiple attack types and volumes are launched simultaneously towards a victim. Multi-vector attacks are harder to detect and defend. Multiple and simultaneous mitigation techniques are needed to defeat such attack campaigns. It is also common for attackers to change attack vectors right after a successful mitigation, burdening their opponents with changing their defense methods.

The ultimate conclusion derived from these real scenarios is that modern attacks detection and mitigation are most certainly complicated and highly convoluted tasks. They demand a comprehensive knowledge of the attack attributes, the targeted normal behavior/traffic patterns, as well as the attacker's on-going and past actions. Even more challenging, retrieving all the analytics needed for detecting these attacks is not simple to obtain with the industry's current capabilities.

The DOTS signal channel protocol [I-D.ietf-dots-signal-channel] is used to carry information about a network resource or a network (or a part thereof) that is under a Distributed Denial of Service (DDoS) attack. Such information is sent by a DOTS client to one or multiple DOTS servers so that appropriate mitigation actions are undertaken on traffic deemed suspicious. Various use cases are discussed in [I-D.ietf-dots-use-cases].

Typically, DOTS clients can be integrated within a DDoS attack detector, or network and security elements that have been actively engaged with ongoing attacks. The DOTS client mitigation environment determines that it is no longer possible or practical for it to handle these attacks. This can be due to lack of resources or security capabilities, as derived from the complexities and the intensity of these attacks. In this circumstance, the DOTS client has invaluable knowledge about the actual attacks that need to be handled by the DOTS server. By enabling the DOTS client to share this comprehensive knowledge of an ongoing attack under specific circumstances, the DOTS server can drastically increase its abilities to accomplish successful mitigation. While the attack is being handled by the DOTS server associated mitigation resources, the DOTS server has the knowledge about the ongoing attack mitigation. The DOTS server can share this information with the DOTS client so that the client can better assess and evaluate the actual mitigation realized.

In some deployments, DOTS clients can send mitigation hints derived from attack details to DOTS servers, with the full understanding that the DOTS server may ignore mitigation hints, as described in [RFC8612] (Gen-004). Mitigation hints will be transmitted across the DOTS signal channel, as the data channel may not be functional during an attack. How a DOTS server is handling normal and attack traffic attributes, and mitigation hints is implementation-specific.

Both DOTS client and server can benefit this information by presenting various information in relevant management, reporting, and portal systems.

This document defines DOTS telemetry attributes the DOTS client can convey to the DOTS server, and vice versa. The DOTS telemetry attributes are not mandatory fields. Nevertheless, when DOTS telemetry attributes are available to a DOTS agent, and absent any policy, it can signal the attributes in order to optimize the overall mitigation service provisioned using DOTS. Some of the DOTS telemetry data are not shared during an attack time.

2. Terminology

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

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

"DOTS Telemetry" is defined as the collection of attributes that are used to characterize normal traffic baseline, attacks and their mitigation measures, and any related information that may help in enforcing countermeasures. The DOTS Telemetry is an optional set of attributes that can be signaled in the DOTS signal channel protocol.

The meaning of the symbols in YANG tree diagrams is defined in [RFC8340].

3. DOTS Telemetry: Overview & Purpose

When signaling a mitigation request, it is most certainly beneficial for the DOTS client to signal to the DOTS server any knowledge regarding ongoing attacks. This can happen in cases where DOTS clients are asking the DOTS server for support in defending against attacks that they have already detected and/or mitigated. These actions taken by DOTS clients are referred to as "signaling the DOTS Telemetry".

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

A basic requirement of security operation teams is to be aware and get visibility into the attacks they need to handle. The DOTS server security operation teams benefit from the DOTS telemetry, especially from the reports of ongoing attacks. Even if some mitigation can be automated, operational teams can use the DOTS telemetry to be prepared for attack mitigation and to assign the correct resources (operation staff, networking and mitigation) for the specific service. Similarly, security operation personnel at the DOTS client side ask for feedback about their requests for protection. Therefore, it is valuable for the DOTS server to share DOTS telemetry with the DOTS client. Thus mutual sharing of information is crucial for "closing the mitigation loop" between the DOTS client and server. For the server side team, it is important to realize that the same attacks that the DOTS server's mitigation resources are seeing are those that the DOTS client is asking to mitigate. For the DOTS client side team, it is important to realize that the DOTS clients receive the required service. For example: understanding that "I asked for mitigation of two attacks and my DOTS server detects and mitigates only one...". Cases of inconsistency in attack

classification between DOTS client and server can be high-lighted, and maybe handled, using the DOTS telemetry attributes.

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

If the DOTS server's mitigation resources have the capabilities to facilitate the DOTS telemetry, the DOTS server adopts its protection strategy and activates the required countermeasures immediately (automation enabled). The overall results of this adoption are optimized attack mitigation decisions and actions.

The DOTS telemetry can also be used to tune the DDoS mitigators with the correct state of the attack. During the last few years, DDoS attack detection technologies have evolved from threshold-based detection (that is, cases when all or specific parts of traffic cross a pre-defined threshold for a certain period of time is considered as an attack) to an "anomaly detection" approach. In anomaly detection, the main idea is to maintain rigorous learning of "normal" behavior and where an "anomaly" (or an attack) is identified and categorized based on the knowledge about the normal behavior and a deviation from this normal behavior. Machine learning approaches are used such that the actual "traffic thresholds" are "automatically calculated" by learning the protected entity normal traffic behavior during peace time. The normal traffic characterization learned is referred to as the "normal traffic baseline". An attack is detected when the victim's actual traffic is deviating from this normal baseline.

In addition, subsequent activities toward mitigating an attack are much more challenging. The ability to distinguish legitimate traffic from attacker traffic on a per packet basis is complex. This complexity originates from the fact that the packet itself may look "legitimate" and no attack signature can be identified. The anomaly can be identified only after detailed statistical analysis. DDoS attack mitigators use the normal baseline during the mitigation of an attack to identify and categorize the expected appearance of a specific traffic pattern. Particularly the mitigators use the normal baseline to recognize the "level of normality" needs to be achieved during the various mitigation process.

Normal baseline calculation is performed based on continuous learning of the normal behavior of the protected entities. The minimum learning period varies from hours to days and even weeks, depending on the protected application behavior. The baseline cannot be learned during active attacks because attack conditions do not characterize the protected entities' normal behavior.

If the DOTS client has calculated the normal baseline of its protected entities, signaling this attribute to the DOTS server along with the attack traffic levels is significantly valuable. The DOTS server benefits from this telemetry by tuning its mitigation resources with the DOTS client's normal baseline. The DOTS server mitigators use the baseline to familiarize themselves with the attack victim's normal behavior and target the baseline as the level of normality they need to achieve. Consequently, the overall mitigation performances obtained are dramatically improved in terms of time to mitigate, accuracy, false-negative, false-positive, and other measures.

Mitigation of attacks without having certain knowledge of normal traffic can be inaccurate at best. This is especially true for recursive signaling (see Section 3.2.3 in [I-D.ietf-dots-use-cases]). In addition, the highly diverse types of use-cases where DOTS clients are integrated also emphasize the need for knowledge of client behavior. Consequently, common global thresholds for attack detection practically cannot be realized. Each DOTS client can have its own levels of traffic and normal behavior. Without facilitating normal baseline signaling, it may be very difficult for DOTS servers in some cases to detect and mitigate the attacks accurately. It is important to emphasize that it is practically impossible for the server's mitigators to calculate the normal baseline, in cases they do not have any knowledge of the traffic beforehand. In addition, baseline learning requires a period of time that cannot be afforded during active attack. Of course, this information can be provided using out-of-band mechanisms or manual configuration at the risk to maintain inaccurate information as the network evolves and "normal" patterns change. The use of a dynamic and collaborative means between the DOTS client and server to identify and share key parameters for the sake of efficient DDoS protect is valuable.

During a high volume attack, DOTS client pipes can be totally saturated. The DOTS client asks the DOTS server to handle the attack upstream so that DOTS client pipes return to a reasonable load level (normal pattern, ideally). At this point, it is essential to ensure that the DOTS server does not overwhelm the DOTS client pipes by sending back "clean traffic", or what it believes is "clean". This can happen when the server has not managed to detect and mitigate all the attacks launched towards the client. In this case, it can be valuable to clients to signal to server the "Total pipe capacity", which is the level of traffic the clients can absorb from the upstream server. Dynamic updating of the condition of pipes between DOTS agents while they are under a DDoS attack is essential. For example, for cases of multiple DOTS clients share the same physical connectivity pipes. It is important to note, that the term "pipe" noted here does not necessarily represent physical pipe, but rather

represents the current level of traffic client can observe from server. The server should activate other mechanisms to ensure it does not saturate the client's pipes unintentionally. The rate-limit action defined in [I-D.ietf-dots-data-channel] can be a reasonable candidate to achieve this objective; the client can ask for the type of traffic (such as ICMP, UDP, TCP port 80) it prefers to limit.

To summarize, timely and effective signaling of up-to-date DOTS telemetry to all elements involved in the mitigation process is essential and absolutely improves the overall service effectiveness. Bi-directional feedback between DOTS agents is required for the increased awareness of each party, supporting superior and highly efficient attack mitigation service.

4. DOTS Telemetry Attributes

This section outlines the set of DOTS telemetry attributes. The ultimate objective of these attributes is to allow for the complete knowledge of attacks and the various particulars that can best characterize attacks.

The description and motivation behind each attribute were presented in Section 3. DOTS telemetry attributes are optionally signaled and therefore MUST NOT be treated as mandatory fields in the DOTS signal channel protocol.

4.1. Pre-mitigation DOTS Telemetry Attributes

The following pre-mitigation telemetry attributes can be signaled from the DOTS client to the DOTS server.

- o DISCUSSION NOTES: (1) Some telemetry can be communicated using DOTS data channel. (2) Evaluate the risk of fragmentation, or (3) check if we can define a dedicated URI for telemetry (e.g.: use ./telemetry). Some of the information is not specific to each mitigation request.

4.1.1. Total Traffic Normal Baseline

The low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile) and peak values of "Total traffic normal baselines" measured in kilobytes per second or megabytes per second or gigabytes per second.

4.1.2. Total Pipe Capability

The limit of traffic volume, in kilobytes per second or megabytes per second or gigabytes per second. This attribute represents the DOTS client domain pipe limit.

- o NOTE: Multi-homing case to be considered.

4.1.3. Total Attack Traffic

The low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile) and peak values of total attack traffic measured in kilobytes per second or megabytes per second or gigabytes per second.

4.1.4. Total Traffic

The low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile) and peak values of total traffic during a DDoS attack measured in kilobytes per second or megabytes per second or gigabytes per second.

4.1.5. Attack Details

Various information and details that describe the on-going attacks that needs to be mitigated by the DOTS server. The attack details need to cover well-known and common attacks (such as a SYN Flood) along with new emerging or vendor-specific attacks. The following fields describing the on-going attack are discussed:

vendor-id: Vendor ID is a security vendor's Enterprise Number as registered with IANA [Enterprise-Numbers]. It is a four-byte integer value.

This is a mandatory sub-attribute.

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

This is a mandatory sub-attribute.

attack-name: Textual representation of attack description. Natural Language Processing (e.g., word embedding) can possibly be used to map the attack description to an attack type. Textual representation of attack solves two problems (a) avoids the need to create mapping tables manually between vendors (2) Avoids the need to standardize attack types which keep evolving.

This is a mandatory sub-attribute

attack-severity: Attack severity. Emergency (0), critical (1) and alert (2).

This is an optional sub-attribute

4.2. DOTS Client to Server Mitigation Efficacy DOTS Telemetry Attributes

The mitigation efficacy telemetry attributes can be signaled from the DOTS client to the DOTS server as part of the periodic mitigation efficacy updates to the server.

4.2.1. Total Attack Traffic

The low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile) and peak values of total attack traffic the DOTS client still sees during the active mitigation service measured in kilobytes per second or megabytes per second or gigabytes per second.

4.2.2. Attack Details

The overall attack details as observed from the DOTS client perspective during the active mitigation service. The same attributes defined in Section 4.1.5 are applicable here.

4.3. DOTS Server to Client Mitigation Status DOTS Telemetry Attributes

The mitigation status telemetry attributes can be signaled from the DOTS server to the DOTS client as part of the periodic mitigation status update.

4.3.1. Mitigation Status

As defined in [RFC8612], the actual mitigation activities can include several countermeasure mechanisms. The DOTS server SHOULD signal the current operational status to each relevant countermeasure. A list of attacks detected by each countermeasure. The same attributes defined for Section 4.1.5 are applicable here for describing the attacks detected and mitigated.

5. DOTS Telemetry YANG Module

5.1. Tree Structure

TODO

5.2. YANG Module

TODO

6. IANA Considerations

6.1. DOTS Signal Channel CBOR Mappings Registry

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

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

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

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

6.2. DOTS Signal Telemetry YANG Module

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC7950] within the "YANG Parameters" registry.

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

7. Security Considerations

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

8. Acknowledgements

The authors would like to thank Flemming Andreassen, Liang Xia, and Kaname Nishizuka co-authors of <https://tools.ietf.org/html/draft-doron-dots-telemetry-00> draft and everyone who had contributed to that document.

9. References

9.1. Normative References

- [Enterprise-Numbers]
"Private Enterprise Numbers", 2005, <<http://www.iana.org/assignments/enterprise-numbers.html>>.
- [I-D.ietf-dots-data-channel]
Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-30 (work in progress), July 2019.
- [I-D.ietf-dots-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-35 (work in progress), July 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
Open Threat Signaling", draft-ietf-dots-use-cases-17 (work
in progress), January 2019.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
<<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8612] Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
Threat Signaling (DOTS) Requirements", RFC 8612,
DOI 10.17487/RFC8612, May 2019,
<<https://www.rfc-editor.org/info/rfc8612>>.

Authors' Addresses

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

Email: kondtir@gmail.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Ehud Doron
Radware Ltd.
Raoul Wallenberg Street
Tel-Aviv 69710
Israel

Email: ehudd@radware.com