DOTS F. Andreasen
Internet Draft T. Reddy
Intended status: Standards Track Cisco
Expires: April 30, 2017 October 31, 2016

Distributed Denial-of-Service Open Threat Signaling (DOTS) Information and Data Model
draft-andreasen-dots-info-data-model-01.txt

Abstract

This document defines an information model and a data model for Distributed Denial-of-Service Open Threat Signaling (DOTS). The document specifies the Message and Information Elements that are transported between DOTS agents and their interconnected relationships. The primary purpose of the DOTS Information and Data Model is to address the DOTS requirements and DOTS use cases.

Status of this Memo

This Internet-Draft will expire on March 30, 2017.

Copyright Notice

Table of Contents

1. Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale 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 client, a router, a firewall, or an entire network.

In order to mitigate a DDoS attack while still providing service to legitimate entities, it is necessary to identify and separate the majority of attack traffic from legitimate traffic and only forward the latter to the entity under attack, which is also known as "scrubbing". Depending on the type of attack, the scrubbing process may be more or less complicated, and in some cases, e.g. a bandwidth saturation, it must be done upstream of the DDoS attack target.

DDoS Open Threat Signaling (DOTS) defines an architecture to help solve these issues (see [I-D.ietf-dots-architecture]). In the DOTS architecture, a DDoS attack target is associated with a DOTS client which can signal a DOTS server for help in mitigating an attack. The DOTS client and DOTS server (collectively referred to as DOTS agents) can interact with each other over two different channels: a signal and a data channel, as illustrated in Figure 1.

```
 +---------------+                                 +---------------+
 |               | <------- Signal Channel ------> |               |
 |  DOTS Client  |                                 |  DOTS Server  |
 |               | <=======  Data Channel  ======> |               |
 +---------------+                                 +---------------+
```

: DOTS signal and data channels

The DOTS signal channel is primarily used to convey information related to a possible DDoS attack so appropriate mitigation actions can be undertaken on the suspect traffic. The DOTS data channel is used for infrequent bulk data exchange between DOTS agents in the aim to significantly augment attack response coordination.

In this document, we define the overall information model and data model for the DOTS signal channel and data channel. The information and data models are designed to adhere to the overall DOTS architecture [I-D.ietf-dots-architecture] , the DOTS use case scenarios, and the DOTS requirements [I-D.ietf-dots-requirements].

2. Notational Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The reader should be familiar with the terms defined in [I-D.ietf-dots-architecture].

3. Information Model

The information model is broken into the following areas:

- General, which provides a general high-level overview of the overall information model and its structure

- Signal Channel specific, which provides an API-style interface description for the signal channel

- Data Channel specific, which provides an API-style interface description for the data channel

- Information Element specific, which describes the various information elements used by the above

3.1. General

DOTS supports request/response style interactions as well as asynchronous messages as illustrated below:

```
DOTS-Request(parameters) => DOTS-Response(parameters)

DOTS-Notification(parameters)
```

DOTS requests, responses and notifications are collectively referred to as DOTS messages, and associated with each DOTS message is one or more parameters. DOTS messages include one or more parameters, each of which belong to one of the following categories

- Mandatory, which are base DOTS protocol parameters that MUST always be included with the message in question.

- Optional, which are base DOTS protocol parameters that MAY be included with the message in question.

- Extensions, which are extended DOTS protocol parameters that MAY be included with the message in question.

Mandatory and optional DOTS protocol parameters MUST be supported by all DOTS agents. Extension DOTS protocol parameters define extended functionality above and beyond the base DOTS protocol that MAY be supported by a particular DOTS agent. The DOTS protocol includes an extension framework that enables each DOTS agent to determine which extension a peer DOTS agent supports. The extension framework also enables a DOTS agent to specify how to handle any unsupported extensions.

DOTS supports versioning in the form of a DOTS protocol version, which is included in every DOTS message. If a DOTS agent receives a DOTS request with an unsupported protocol version, it will reply with a failure and an indication of the protocol version(s) supported. This enables a future version of the DOTS protocol to be defined in a backwards compatible manner. If a DOTS Notification with an unsupported protocol version is received, it will simply be discarded.

All DOTS messages include a message-id and an agent-id. The agent-id identifies the originator of the message, whereas the message-id provides an identifier for DOTS Request/Response correlation and DOTS Notification identification.

- [Editor’s note: There are different views on whether an agent-id is needed at the DOTS level or whether the underlying security mechanisms (incl. authenticated identity via (D)TLS) suffices. For now, the draft includes an explicit agent-id at the DOTS level].

- [Editor’s note: Need to look more closely at how we ensure message-id uniqueness. One option is to include the agent-id from the request or notification as part of it. Also, not clear (yet) if this is really information/data model or protocol]

DOTS messages are exchanged over the DOTS Signal channel and the DOTS Data Channel. In either case, there is a channel establishment procedure taking place initially whereby:

1. The DOTS client determines the address of a DOTS server to establish the DOTS channel with. The base DOTS protocol assumes the DOTS client has already obtained either this address, or a domain name [RFC1034] or DNS SRV name [RFC2782] that can lead to it. One way of achieving that is by provisioning.

2. The DOTS agents mutually authenticate each other. The Information and Data model assumes that the DOTS protocol will run on top of a secure transport (e.g. TLS [RFC5246] or DTLS [RFC6347]) that also provides mutual authentication. The DOTS agent-id will need to be tied-to or covered by this transport-level mutual authentication. If not, explicit mutual authentication of the DOTS agent-id at the DOTS protocol level will be needed

   - [Editor's note: The mutual authentication aspect of the DOTS agent-id is a bit sketchy; needs more consideration]

Note that the mutual authentication of the DOTS agents are needed to verify the service relationship between the DOTS client and server as well as govern relevant authorization policies with respect further DOTS operation (e.g. scrubbing of traffic for the client in question).

- [Editor's note: The below is work in progress - main focus is on overall approach and messages right now. The actual parameters are expected to evolve in future versions]

3.2. Signal Channel Messages

In this section, we describe the signal channel messages.

- [Editor's note: Most (all ?) messages can be extended - currently not clearly shown in the following]

3.2.1. Open Signal Channel

The OpenSignalChannel request establishes a signal channel from the LocalAgent to the RemoteAgent:

```
OSCreply OpenSignalChannel(LocalAgentId, RemoteAgent
                          [, ExtensionsSupported])
```

LocalAgentId is the local DOTS client id.

RemoteAgent is one or more of IP-address(es), domain-name and DNS SRV name for the remote DOTS agent with which the signal channel is to be established. When more than one of these is present, the priority order for client use is DNS SRV, domain-name and IP-address(es). A lower priority MUST NOT be used unless a higher priority fails to produce a response.

o Note that the intent of this priority order is not to replace any DNS caching but rather to provide a "last resort" in case of DNS failure. If domain name use is not desired, then RemoteAgent MUST only include IP-address(es).

ExtensionsSupported may optionally be included to indicate which extensions are supported by the DOTS client.

A successful OSCreply contains the following information:

o Response code, which indicates the outcome of the request

o RemoteAgentId, which is the peer DOTS agent Id.

o SignalChannel, which is a handle for the new signal channel assuming the request succeeded

o [optional] ExtensionsSupported, which lists the extensions supported by the remote DOTS agent.

A failure OSCreply contains the following information:

o Response code, which indicates the outcome of the request

o RemoteAgent, which indicates a different agent to establish the signal channel with (redirect)

o [optional] ExtensionsSupported, which lists the extensions supported by the remote DOTS agent.

3.2.2. Close Signal Channel

The CloseSignalChannel request closes a previously established signal channel:

```
CSCreply CloseSignalChannel(SignalChannel)
```

SignalChannel is the handle for the signal channel to be closed.

The CSCreply contains the following information

- Response code, which indicates the outcome of the request

3.2.3. Redirect Signal Channel

The RedirectSignalChannel request instructs the peer DOTS agent to change the signal channel to the alternative DOTS agent indicated.

- [Editor’s note: At the IETF Berlin meeting, there was discussion around using anycast to possibly avoid redirection. Anycast however would not be mandatory, and even when supported, it may be desirable to "redirect" to a non-anycast address after initial discovery for stability]

- [Editor’s note: Data channel is currently associated with the data channel. Either they both have to get redirected or we need a way of (re)associating the data channel with the new signal channel; maybe a MoveSignalChannel() ?]

- [Editor’s note: When redirecting the channel, do we redirect the existing one or create a new one and close the old one ? The latter seems cleaner, albeit more explicit.]

```
RSCreply RedirectSignalChannel(SignalChannel, NewAgent)
```

SignalChannel is the handle for the signal channel to be redirected.

NewAgent is an IP-address, domain-name or DNS SRV name for the new remote DOTS agent with which the signal channel is to be redirected to.

The CSCreply contains the following information

- Response code, which indicates the outcome of the request. Note that a success response merely indicates successful receipt and reply to the request; successful redirection of signal channel is not implied.

3.2.4. Request Status Update

The RequestStatusUpdate message is a notification to the peer agent asking it to provide a status update as indicated

```
RequestStatusUpdate(Target [, Heartbeat] [, Health] [, Attack]
                              [, Mitigation] [, Efficacy])
```

Target specifies the attack target for which status updates are requested

- [Editor’s note: Need to come up with a more detailed model for how we identify and describe targets]

Heartbeat requests a status update at the heartbeat interval specified.

Health requests health information for the target.

Attack requests attack information for the target.

Mitigation requests mitigation status information for the target.

Efficacy requests mitigation efficacy information for the target.

3.2.5. Status Update

The StatusUpdate message is a notification to the peer agent. StatusUpdate provides heartbeat functionality and updates around health, attack status, mitigation status and mitigation efficacy

```
StatusUpdate(Target, [, HealthStatus] [, AttackStatus]
               [, MitigationStatus] [, MitigationEfficacy])
```

Target specifies the attack target for which status updates are provided.

HealthStatus provides overall health for the target

AttackStatus provides information about an ongoing attack for the target

- o [Editor’s note: Do we need to identify and refer to specific attacks for a given target or just the target itself. For now, assume just the target itself. Question applies to other parameters here as well.]

MitigationStatus provides information about current mitigation(s) in place for the target. The status reflects information from the mitigator’s point of view.

MitigationEfficacy provides information about the efficacy of the mitigation. The efficacy reflects information from the attack target’s point of view.

3.2.6. Request Mitigation

The RequestMitigation message is a notification to the peer agent to invoke mitigation for the attack target specified. If the request is received and honored, mitigation will commence and a StatusUpdate message will be sent to reflect that. Note that either message is especially susceptible to loss during an active DDoS attack

```
RequestMitigation(Target)
```

Target specifies the attack target for mitigation is requested.

3.2.7. Stop Mitigation

The StopMitigation message is a notification to the peer agent to stop further attack mitigation for the target indicated. The message is sent on behalf of the attack target towards the mitigator.

```
StopMitigation(Target)
```

Target specifies the attack target for which mitigation is to be stopped.

3.3. Data Channel Messages

In this section, we describe the data channel messages.

o [Editor’s note: Most (all ?) messages can be extended - currently not clearly shown in the following]

3.3.1. Open Data Channel

The OpenDataChannel request opens a Data Channel to be used in conjunction with a previously established Signal Channel

```
ODCreply OpenDataChannel(SignalChannel [, ExtensionsSupported])
```

SignalChannel is the handle for the existing signal channel.

ExtensionsSupported may optionally be included to indicate which extensions are supported by the DOTS client.

A successful ODCreply contains the following information

o DataChannel, which is a handle for the new data channel

o [optional] ExtensionsSupported, which lists the extensions supported by the remote DOTS agent.

A failure ODCreply contains the following information

- Response code, which indicates the outcome of the request

- RemoteAgent, which indicates a different agent to establish the data channel with (redirect)

3.3.2. Close Data Channel

The CloseDataChannel request closes a previously established data channel:

```
CDCreply CloseDataChannel(DataChannel)
```

DataChannel is the handle for the data channel to be closed.

The CDCreply contains the following information

- Response code, which indicates the outcome of the request

3.3.3. Redirect Data Channel

- [Editor’s note: Resolve open questions in Redirect Signal Channel first]

3.3.4. SendData

The SendData request sends data to the peer DOTS agent over the data channel.

```
SDreply SendData(DataChannelData)
```

DataChannelData may contain one or more of the following:

- Blacklists, whitelists, filters, aliases\names)

[Editor’s note: The above needs much more work and overall structure. Break up into individual pieces and make sure it’s modular, extensible and we have clarity on which data is mandatory versus optional to support]

3.4. Information Elements

- [Editor’s note: The following should merely be considered as a bucket of possible IEs at this point; further work needed, including reconciling with message definitions in sections above]

3.4.1. Protocol version

3.4.2. Attack Target

- [Editor’s note: may be superfluous given Mitigation Scope below"]

3.4.3. Agent Id

(identity for each DOTS client and server, contains a least a domain portion that can be authenticated)

3.4.4. Blacklist

(define, retrieve, manage and refer to)

3.4.5. Whitelist

(define, retrieve, manage and refer to)

3.4.6. Attack telemetry

(collected behavioral characteristics defining the nature of a DDoS attack)

- [9/27: Probably extended functionality.]

3.4.7. Mitigator feedback

(attack mitigation feedback from server to client, incl. mitigation status [start, stop, metrics, etc.], attack ended and information about mitigation failure)

3.4.8. Mitigation efficacy

(attack mitigation efficacy feedback from client to server)

3.4.9. Mitigation failure

(unsupported target type, mitigation capacity exceeded, excessive "clean traffic", out-of-service, etc.)

3.4.10. Mitigation Scope

Classless Internet Domain Routing (CIDR) prefixes, BGP prefix, URI, DNS names, E.164, "resource group alias", port range, protocol, or service

- [Editor’s note: comes from requirements - not clear how "protocol" and "service" are defined. Also, consider which URI schemes]

- [Editor’s note: It would probably be useful to structure mitigation scope and related information (like telemetry, blacklist, etc.) into different "types", since different types of targets will have different parameters and different DOTS servers may support different types of attack targets]

Information about the attack (e.g. targeted port range, protocol or scope)

- [Editor’s note: Not clear this is really different from "Mitigation Scope" below - taken from requirement OP-006]

- [9/27: Roland doesn’t think we need this as baseline information.]

3.4.11. Mitigation Scope Conflict

Nature and scope of conflicting mitigation requests received from two or more clients

3.4.12. Resource Group Alias

(define in data channel, refer to in signal/data channel; aliases for mitigation scope)

3.4.13. Mitigation duration

(desired lifetime - renewable)

3.4.14. Peer health

(? - measure of peer health)

3.4.15. Filters

3.4.16. Filter-actions

(rate-limit, discard)

3.4.17. Acceptable signal lossiness

(for unreliable delivery)

3.4.18. Heartbeat interval

3.4.19. Data Channel Address

- Editor’s note: For discussion (not entirely aligned with current architecture draft text); assumes establish signal channel first and learn data channel address through it (would be useful for redirection as well and makes it easier for signal and data channel to terminate on different entities)]

3.4.20. Extensions

(standard, vendor-specific, supported)

4. Data Model

TODO

5. IANA Considerations

TODO

6. Security Considerations

TODO

7. Acknowledgements

TODO

This document was prepared using 2-Word-v2.0.template.dot.

8. References

8.1. Normative References

[RFC1034] Mockapetris, P.V., "Domain Names - concept and facilities", RFC 1034, November 1987.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-dots-architecture] Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-00 (work in progress), July 2016.

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-02 (work in progress), July 2016.

8.2. Informative References

[RFC5246] Dierks, T., and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008

[RFC6347] Rescorla, E., and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012

Authors' Addresses

Flemming Andreasen
Cisco Systems, Inc.
USA

Email: fandreas@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

DOTS
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

E. Doron
Radware Ltd.
T. Reddy
F. Andreasen
Cisco Systems, Inc.
L. Xia
Huawei
K. Nishizuka
NTT Communications

October 30, 2016

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry Specifications
draft-doron-dots-telemetry-00

Abstract

This document aims to enrich DOTS Signaling with various telemetry attributes allowing optimal DDoS/DoS attack mitigation. The nature of the DOTS architecture is to allow DOTS Agents to be integrated in highly diverse environments. Therefore, the DOTS architecture imposes a significant challenge in delivering optimal mitigation services. The DOTS Telemetry covered in this document aims to provide all needed attributes and feedback signaled from DOTS Agents such that optimal mitigation services can be delivered based on DOTS Signaling.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DOTS signaling architecture (see [I-D.ietf-dots-architecture]) is designed to allow anti-DoS services for a vast number of networking, security and operational scenarios aimed to operate in diverse environments. This is a multi-dimensional challenge DOTS needs to meet in order to provide all the signaling requirements as derived from each environment’s unique characteristics. The DOTS Client can be integrated within various elements with large diversity on their security capabilities. In a simple use case, the DOTS Client can be integrated in entities with a very basic understanding of the current security conditions, for example a customer portal with a user that is just realizing that something is "going wrong" with his service but is not aware of the main cause of the service degradation. Here, the DOTS Client can basically signal the need for mitigation along with its identification attributes. In a more advanced use case, the DOTS Client can be integrated within DDoS/DoS attack mitigators (and their control and management environments) 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, the DOTS Server can dramatically 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 comprehend and evaluate the actual mitigation realized. Both DOTS Client and DOTS Server can benefit this information by presenting various information in relevant management, reporting and portal systems.

"DOTS Telemetry" is defined as the collection of attributes characterizing the actual attacks that have been detected and mitigated. The DOTS Telemetry is an optional set of attributes that can be signaled in the various DOTS protocol messages. The DOTS Telemetry can be optionally sent from the DOTS Client to Server and vice versa.

This document aims to define all the required DOTS Telemetry attributes in order to use DOTS Signal and Data Channels for DOTS Telemetry signaling. Due to the diversity of environments DOTS Agents are designed to be integrated within, the DOTS Telemetry attributes (all of them as a whole, or some of them) are not mandatory fields in any type of DOTS protocol message. Nevertheless, when DOTS Telemetry attributes are available to the DOTS Agent it MAY signal the attributes in order to optimize the overall service provisioned using DOTS. Other basic minimum set of DOTS mandatory signaling attributes (like "targeted entity", Targeted IP address and so on), that are covered in other DOTS documents, are not reiterated in this document. No assumption is made regarding the DOTS Telemetry’s actual collection methodology.

The document is divided into three logical parts: The first outlines the need for DOTS Telemetry. The second covers the actual telemetry attributes needed for providing comprehensive mitigation services. The third describes the telemetry attributes needed for each of the DOTS Signaling stages. Several typical use cases are also discussed in detail.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Definition of Terms

This document uses the various terms defined in DOTS reuirements document, see [I-D.ietf-dots-requirements].

2. Using the DOTS telemetry for a successful mitigation

The cyber security battle between the adversary and security countermeasures is an everlasting fight. The DoS/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 DoS/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, FW, transit links, and so on) from serving legitimate user traffic. Here the main method of the attackers is to send a large volume or high PPS of traffic toward the victim’s infrastructure. 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 ports open at network edges can be attractive attack targets. Application layer attacks are considered more complex and hard to categorize, therefore harder to detect and mitigate efficiently.

To compound the problem, attackers also leverage multi-vectored attacks. These merciless attacks are assembled from dynamic attack vectors (Network/Application) and tactics. Here, multiple attack vectors formed by multiple attack types and volumes are launched simultaneously towards the 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 only moments after a successful mitigation, burdening their opponents with changing their defense methods.

The ultimate conclusion derived from these real-life 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.

With all this in mind, 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 Agent are referred to as "signaling the DOTS Telemetry". If attacks are already detected and categorized, the DOTS Server, and his 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. Therefore, the DDOS mitigation service must treat the ongoing attack details from the Client as hints, and cannot completely rely or trust the attack details conveyed by the DOTS client. Nevertheless, the DOTS Telemetry should support the identification of such misalignment conditions.

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. They 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 Client and Server. For the Server side teams, it is important to realize that "the same attacks that I am seeing are those that my client is asking me to mitigate?." For the Client side, it is important to realize that the Clients receive the required service. For example: understanding that "I asked for mitigation of two attacks and my 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 various attributes.

In addition, management and orchestration systems, at both Client and Server side, can potentially use DOTS Telemetry as a feedback to automate various control and management activities derived from ongoing information signaled.

Should the DOTS Server’s mitigation resources have the capabilities to facilitate the DOTS Telemetry, the Server adopts its protection strategy and activates the required countermeasures immediately. The overall results of this adoption are optimized attack mitigation decisions and actions.

The DOTS Telemetry can also be used to tune the mitigators with the correct state of the attack. During the last few years, DDoS/DoS 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, the subsequent activities toward mitigating the attack are much more challenging. The ability to distinguish legitimate traffic from attacker traffic on a per packet basis is complex. This complexity originates in 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/DoS attack mitigators use the normal baseline during the actual 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 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 DOTS environments where it is assumed that there is no universal DDoS attack scale threshold triggering an attack across administrative domains (see [I-D.ietf-dots-architecture]). 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 attacks detection practically cannot be realized. Each client can have his own levels of traffic and normal behavior. Without facilitate baseline signaling, it can be very difficult for Server 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.

As mentioned above, the task of isolating legitimate from attacker traffic is extremely difficult to achieve. A common mechanism that DDoS/DoS mitigators use to achieve such a distinction is to authenticate source IP addresses that send traffic towards protected entities. The source IP address can be authenticated as legitimate or as a malicious BOT. Traffic from a BOT can be discarded or can be rate-limited. Authentication can be performed using various techniques; actively sending various challenges towards source IP addresses is a common method. SYN Cookies, CAPTCHA, cryptographic puzzle and others are examples of challenge-response tests used by mitigators to determine whether the user is legitimate or a BOT. Most certainly, building a list of authenticated source IP addresses is a task that consumes resources and takes a long period of time to construct. If the DOTS Client has already built a list of authenticated IP addresses, the DOTS Server can use this list to safely serve these IP addresses without any further need to re-authenticate them. It is important to mention that "authenticated IPs" are different from IP addresses in a "white list". This is mainly because the authenticated IPs addresses are not predefined and are not known upfront to the DOTS Agents. In addition, a source IP address is treated as an authenticated IP address for a limited period of time.

During a high volume attack, DOTS Client pipes can be totally saturated. The Client asks the Server to handle the attack upstream so that DOTS Client pipes return to a reasonable load level. 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 essentially. For example, for cases of multiple DOTS Clients share the same physical connectivity pipes. It is important to note, that the term "pipe" noted here does not necessary represent physical pipe, but rather represents the current level of traffic Client can observe from Server. The Server should activate other mechanisms to ensure it does not saturate the Client’s pipes unintentionally. The Rate Limiter 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 elements is required for the increased awareness of each party, supporting superior and highly efficient attack mitigation service.

3. 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. This section presents the attributes required for each stage of the DOTS Signaling protocol (see [I-D.reddy-dots-signal-channel] and [I-D.nishizuka-dots-inter-domain-mechanism]). Other way of using telemetry attributes is allowing DOTS Server to receive relevant DOTS Telemetry before the actual attacks are launched using the DOTS Data Channel [I-D.reddy-dots-signal-channel].

The description and motivation behind each attribute were presented in previous sections in this document. The data model and the actual integration within the DOTS Protocol are out of scope of this document. It is expected that the following attributes will be covered in any of the DOTS Protocol and DOTS Data Model standards. As explained in previous sections, the DOTS Telemetry attributes are optionally signaled and therefore SHOULD NOT be treated as mandatory fields in any DOTS protocol messages.

3.1. Pre-mitigation DOTS Telemetry attributes

The Pre-mitigation telemetry attributes MAY be signaled from the DOTS Client to the DOTS Server as part of the initiation of a DOTS service request or during peace time using the DOTS Data Channel. Can be signaled during a "Mitigation Request" (see also [I-D.nishizuka-dots-inter-domain-mechanism]) session, or as part of the "POST request" DOTS Signal (see also [I-D.reddy-dots-signal-channel]). The following attributes are required:

3.1.1. "Normal Baselines" of legitimate traffic

Average, x percentile and peak values of "Total traffic normal baselines". PPS and BPS of the traffic are required.

[[EDITOR’S NOTE: We request feedback from the working group about possible types of baselines to be signaled.]]

3.1.2. "Total Attack Traffic volume"

Current and peak values of "Total attack traffic". PPS and BPS of attack traffic are required.

3.1.3. "Attack Details"

Various information and details that describe the on-going attacks that need to be mitigated by the DOTS Server. The Attack Details need to cover well-known and common attacks (such as a SYN Flood) along with new emerging or vendor-specific attacks. The following is a suggestion for the required fields in the Attack Details (this definition follows the CEF Common Event Formula event definition, see also [CEF] for more description):

Vendor |Version| Attack ID| Attack Name| Attack Severity| Extension

Where Extension is a placeholder for additional fields. Examples for such fields are: Attack Classification, for example UDP port 80, Layer 7 attack signature - Regex with Layer 7 attack signatures. These can be defined as relevant key value pairs. Common and well-known attack IDs SHOULD be standardized, and the vendor-specific IDs SHOULD be specifically defined by each vendor.

The DOTS server should only treat the attack details as hints, and not as a strict attribute to comply to. Please see requirement OP-004 in [I-D.ietf-dots-requirements].

[[EDITOR’S NOTE: We request feedback from the working group about possible alternatives for presenting "Attack Details" and various characteristics of attacks.]]

3.1.4. "Total pipe capacity"

The limit of traffic volume, in BPS and PPS. The DOTS Server SHALL eliminate sending this back as clean traffic. This attribute represents the DOTS Client’s pipe limits.

3.1.5. List of already "Authenticated source IPs"

List of source IP addresses that the DOTS Client has already identified as authenticated IP addresses.

[[EDITOR’S NOTE: We request feedback from the working group about the way to support this kind of IP address list under various NAT strategies deployed in the Client’s network. The same support is required for "white lists" and "black lists" referred to in several DOTS drafts, see [I-D.reddy-dots-data-channel].]]

3.2. Client to Server Mitigation Status DOTS Telemetry attributes

The Mitigation Status telemetry attributes MAY be signaled from the DOTS Client to the DOTS Server as part of the periodic mitigation status update as realized by the Server. This can be signaled during "Mitigation Efficacy Update" (see also [I-D.nishizuka-dots-inter-domain-mechanism] session, or as part of the - "PUT request" DOTS signal (see also [I-D.reddy-dots-signal-channel]).

The following attributes are required:

3.2.1. Current "Total traffic volumes"

Current values of the total traffic, in BPS and PPS, that arrive at the DOTS Client sites. In addition, the Peak and x percentile of traffic, in BPS and PPS, MAY also be signaled.

3.2.2. Current "Total Attack Traffic"

The total attack traffic volume, bps and pps, that the DOTS Client still sees during the active mitigation service. In addition, the Peak and x percentile of traffic, in BPS and PPS, MAY also be signaled.

3.2.3. "Mitigation Efficacy Factor"

A factor defining the overall Mitigation Efficacy from the Client perspective. By way of suggestion, the Mitigation Efficacy Factor can be defined as the current clean traffic ratio to the normal baseline. Network and Application Performance Monitoring attributes can also be considered here.

3.2.4. "Attack Details"

The overall attack details as observed from the DOTS Client perspective. The same data models that will be defined for the Pre-mitigation DOTS Telemetry can also be applicable here.

3.3. Server to Client Mitigation Status DOTS Telemetry attributes

The Mitigation Status telemetry attributes MAY be signaled from the DOTS Server to the DOTS Client as part of the periodic mitigation status update. This can be signaled during "Mitigation Status" (see also [I-D.nishizuka-dots-inter-domain-mechanism] session, or as part of the "GET request" DOTS Signal (see also [I-D.reddy-dots-signal-channel]).

The following attributes are required:

3.3.1. Current "Mitigation Countermeasure status"

As defined in [I-D.ietf-dots-requirements], the actual mitigation activities can include several countermeasure mechanisms. The DOTS Server SHOULD signal the current operational status to each relevant countermeasure. For example: Layer 4 mitigation active/inactive, Layer 7 mitigation active/inactive, and so on. In addition to the status of each countermeasure, the DOTS Server SHOULD also signal: A list of attacks detected by each countermeasure, and the statistics for each countermeasure: Number of bytes/packets for each attack handled, clean traffic volumes, dropped traffic.

It is important to maintain the AttackID among all DOTS communications. The DOTS Client can further use this information for reporting and service fulfillment purposes.

4. DOTS Telemetry Use-cases

DOTS Telemetry can certainly improve numerous DOTS Signaling use cases. Nevertheless, DOTS Telemetry can be most beneficial when dealing with relatively complex use cases where the DOTS Client is integrated into environments with advanced detection and mitigation abilities. In this section, typical use-cases are presented. However, this list of use cases does not eliminate many other scenarios, where the DOTS Telemetry is the pivot in bringing in valuable use cases. It is expected that the DOTS Telemetry notions will be added to the DOTS use cases [I-D.ietf-dots-use-cases] documant.

4.1. Hybrid anti-DoS services use-case

In this common use case, a large enterprise deploys DDoS/DoS mitigators as "in-line" devices on all the enterprise Internet peers. The enterprise’s security and operations teams are aware that in cases of a large volume of attacks their Internet links can get saturated. Therefore, having "in-line" mitigation devices deployed will not help them in maintaining the service level that their organization must maintain. In addition, they understand that they are not capable of operating all the required actions to mitigate multi-vector attacks. For these solid reasons, the enterprise IT decides to purchase MSP (Managed Security Services) for "on demand" DDoS/DoS mitigation services from a MSP Cloud provider. As part of the anti-DoS service delivery, the enterprise and the MSP Cloud provider have agreed to the required SLA. Also, they deploy a DOTS Client at the enterprise premises and the DOTS Server at the MSP cloud. During peace time, the enterprise mitigators build the enterprise protected service’s normal baseline. In cases of attacks that can be mitigated "on-prem", the enterprise is able to deal with the attack with its own resources. Should the attack become a large volume attack and or also become multi-vector, the Internet links of the organization, or even the mitigator links, get saturated. The DOTS Client signals the need for aid in mitigating the on-going attacks from the MSP’s DOTS Server. In order to fulfil his SLA, the MSP uses the DOTS Telemetry it received from the Client to assign the adequate mitigation resources, tune the mitigators with the normal baseline, assign the appropriate personnel to handle the enterprise attacks, and so forth. The enterprise’s security and operations team uses the DOTS Telemetry they received from their DOTS Client to get visibility into the actual mitigation performed "on the cloud" and makes sure the service is fulfilled as expected.

4.2. MSP to MSP anti-DoS services use-case

This use case can be treated as a continuation of the previous use case. The MSP Cloud provider operation team realizes that they have some serious difficulties at their data centers and they are no longer capable of serving the enterprise attacks. A back-to-back DOTS Gateway is implemented at the MSP Cloud to allow redirection of the attack to another MSP Cloud provider for the purpose of attack mitigation. The same processes for using DOTS Telemetry are taken here to ensure continuous service delivery. The same is true for visibility into the actual service provided to the enterprise and to the primary MSP Cloud provider.

5. Acknowledgements

Thanks to Yotam Ben-Ezra and Dennis Usle from Radware for their contribution, careful reading and feedback.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

To Be Added.

8. References

8.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[CEF] ArcSight, Inc., "Common Event Format configuration guide.", 2009, <https://kc.mcafee.com/resources/sites/MCAFEE/content/live/CORP_KNOWLEDGEBASE/78000/KB78712/en_US/CEF_White_Paper_20100722.pdf>.

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-00 (work in progress), July 2016.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-02 (work in progress), July 2016.

[I-D.ietf-dots-use-cases]
Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., and L. Xia, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-01 (work in progress), March 2016.

[I-D.nishizuka-dots-inter-domain-mechanism]
Nishizuka, K., Xia, L., Xia, J., Zhang, D., Fang, L., christopher_gray3@cable.comcast.com, c., and R. Compton, "Inter-domain cooperative DDoS protection mechanism", draft-nishizuka-dots-inter-domain-mechanism-01 (work in progress), July 2016.

[I-D.reddy-dots-data-channel]
Reddy, T., Wing, D., Boucadair, M., Nishizuka, K., and L. Xia, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel", draft-reddy-dots-data-channel-00 (work in progress), August 2016.

[I-D.reddy-dots-signal-channel]
Reddy, T., Boucadair, M., Wing, D., and P. Patil, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel", draft-reddy-dots-signal-channel-01 (work in progress), September 2016.

Authors' Addresses

Ehud Doron
Radware Ltd.
Raoul Wallenberg Street
Tel-Aviv 69710
Israel

Email: ehudd@radware.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Flemming Andreasen
Cisco Systems, Inc.
USA

Email: fandreas@cisco.com

Liang Xia (Frank)
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.xialiang@huawei.com

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

Email: kaname@nttv6.jp

DOTS A. Mortensen
Internet-Draft Arbor Networks, Inc.
Intended status: Informational F. Andreasen
Expires: May 4, 2017 T. Reddy
Cisco Systems, Inc.
C. Gray
Comcast, Inc.
R. Compton
Charter Communications, Inc.
N. Teague
Verisign, Inc.
October 31, 2016

Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
draft-ietf-dots-architecture-01

Abstract

This document describes an architecture for establishing and maintaining Distributed Denial of Service (DDoS) Open Threat Signaling (DOTS) within and between domains. The document does not specify protocols or protocol extensions, instead focusing on defining architectural relationships, components and concepts used in a DOTS deployment.

Status of This Memo

Copyright Notice

Table of Contents

1. Context and Motivation

Signaling the need for help defending against an active distributed denial of service (DDoS) attack requires a common understanding of mechanisms and roles among the parties coordinating defensive response. The signaling layer and supplementary messaging is the focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of coordinating defensive measures among willing peers to mitigate attacks quickly and efficiently, enabling hybrid attack responses coordinated locally at or near the target of an active attack, or anywhere in-path between attack sources and target.

This document describes an architecture used in establishing, maintaining or terminating a DOTS relationship within a domain or between domains.

1.1. Terminology

1.1.1. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.1.2. Definition of Terms

This document uses the terms defined in [I-D.ietf-dots-requirements].

1.2. Scope

In this architecture, DOTS clients and servers communicate using the DOTS signaling. As a result of signals from a DOTS client, the DOTS server may modify the forwarding path of traffic destined for the attack target(s), for example by diverting traffic to a mitigator or pool of mitigators, where policy may be applied to distinguish and discard attack traffic. Any such policy is deployment-specific.

The DOTS architecture presented here is applicable across network administrative domains - for example, between an enterprise domain and the domain of a third-party attack mitigation service - as well as to a single administrative domain. DOTS is generally assumed to be most effective when aiding coordination of attack response between two or more participating network domains, but single domain scenarios are valuable in their own right, as when aggregating intra-domain DOTS client signals for inter-domain coordinated attack response.

This document does not address any administrative or business agreements that may be established between involved DOTS parties. Those considerations are out of scope. Regardless, this document assumes necessary authentication and authorization mechanism are put in place so that only authorized clients can invoke the DOTS service.

1.3. Assumptions

This document makes the following assumptions:

- All domains in which DOTS is deployed are assumed to offer the required connectivity between DOTS agents and any intermediary network elements, but the architecture imposes no additional limitations on the form of connectivity.

- Congestion and resource exhaustion are intended outcomes of a DDoS attack [RFC4732]. Some operators may utilize non-impacted paths or networks for DOTS, but in general conditions should be assumed to be hostile and that DOTS must be able to function in all circumstances, including when the signaling path is significantly impaired.

- There is no universal DDoS attack scale threshold triggering a coordinated response across administrative domains. A network domain administrator, or service or application owner may arbitrarily set attack scale threshold triggers, or manually send requests for mitigation.

- Mitigation requests may be sent to one or more upstream DOTS servers based on criteria determined by DOTS client administrators. The number of DOTS servers with which a given DOTS client has established signaling sessions is determined by local policy and is deployment-specific.

- The mitigation capacity and/or capability of domains receiving requests for coordinated attack response is opaque to the domains sending the request. The domain receiving the DOTS client signal may or may not have sufficient capacity or capability to filter any or all DDoS attack traffic directed at a target. In either case, the upstream DOTS server may redirect a request to another DOTS server. Redirection may be local to the redirecting DOTS server’s domain, or may involve a third-party domain.

- DOTS client and server signals, as well as messages sent through the data channel, are sent across any transit networks with the same probability of delivery as any other traffic between the DOTS client domain and the DOTS server domain. Any encapsulation required for successful delivery is left untouched by transit network elements. DOTS server and DOTS client cannot assume any preferential treatment of DOTS signals. Such preferential treatment may be available in some deployments, and the DOTS architecture does not preclude its use when available. However, DOTS itself does not address how that may be done.

- The architecture allows for, but does not assume, the presence of Quality of Service (QoS) policy agreements between DOTS-enabled peer networks or local QoS prioritization aimed at ensuring delivery of DOTS messages between DOTS agents. QoS is an operational consideration only, not a functional part of the DOTS architecture.

- The signal channel and the data channel may be loosely coupled, and need not terminate on the same DOTS server.

2. Architecture

The basic high-level DOTS architecture is illustrated in Figure 1:

```
   +-----------+            +-------------+
   | Mitigator | ~~~~~~~~~~ | DOTS Server |
   +-----------+            +-------------+
                                   |
                                   |
                                   |
   +---------------+        +-------------+
   | Attack Target | ~~~~~~ | DOTS Client |
   +---------------+        +-------------+
```

Figure 1: Basic DOTS Architecture

A simple example instantiation of the DOTS architecture could be an enterprise as the attack target for a volumetric DDoS attack, and an upstream DDoS mitigation service as the Mitigator. The enterprise (attack target) is connected to the Internet via a link that is getting saturated, and the enterprise suspects it is under DDoS attack. The enterprise has a DOTS client, which obtains information about the DDoS attack, and signals the DOTS server for help in mitigating the attack. The DOTS server in turn invokes one or more mitigators, which are tasked with mitigating the actual DDoS attack, and hence aim to suppress the attack traffic while allowing valid traffic to reach the attack target.

The scope of the DOTS specifications is the interfaces between the DOTS client and DOTS server. The interfaces to the attack target and the mitigator are out of scope of DOTS. Similarly, the operation of both the attack target and the mitigator are out of scope of DOTS.

Thus, DOTS neither specifies how an attack target decides it is under DDoS attack, nor does DOTS specify how a mitigator may actually mitigate such an attack. A DOTS client’s request for mitigation is advisory in nature, and may not lead to any mitigation at all, depending on the DOTS server domain’s capacity and willingness to mitigate on behalf of the DOTS client’s domain.

As illustrated in Figure 2, there are two interfaces between the DOTS server and the DOTS client:

```
+---------------+                                 +---------------+
|               | <------- Signal Channel ------> |               |
|  DOTS Client  |                                 |  DOTS Server  |
|               | <=======  Data Channel  ======> |               |
+---------------+                                 +---------------+
```

Figure 2: DOTS Interfaces

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 signaling sessions by connecting to the provided DOTS server addresses.

[[EDITOR’S NOTE: We request feedback from the working group about the mechanism of server discovery.]]

The primary purpose of the signal channel is for a DOTS client to ask a DOTS server for help in mitigating an attack, and for the DOTS server to inform the DOTS client about the status of such mitigation. The DOTS client does this by sending a client signal, which contains information about the attack target or targets. The client signal may also include telemetry information about the attack, if the DOTS client has such information available. The DOTS server in turn sends a server signal to inform the DOTS client of whether it will honor the mitigation request. Assuming it will, the DOTS server initiates attack mitigation (by means outside of DOTS), and periodically informs the DOTS client about the status of the mitigation. Similarly, the DOTS client periodically informs the DOTS server about the client’s status, which at a minimum provides client (attack target) health information, but it may also include telemetry information about the attack as it is now seen by the client. At some point, the DOTS client may decide to terminate the server-side attack mitigation, which it indicates to the DOTS server over the signal channel. A mitigation may also be terminated if a DOTS client-specified mitigation time limit is exceeded; additional considerations around mitigation time limits may be found below. Note that the signal channel may need to operate over a link that is experiencing a DDoS attack and hence is subject to severe packet loss and high latency.

While DOTS is able to request mitigation with just the signal channel, the addition of the DOTS data channel provides for additional and more efficient capabilities; both channels are required in the DOTS architecture. The primary purpose of the data channel is to support DOTS related configuration and policy information exchange between the DOTS client and the DOTS server. Examples of such information include, but are not limited to:

- Creating identifiers, such as names or aliases, for resources for which mitigation may be requested. Such identifiers may then be used in subsequent signal channel exchanges to refer more efficiently to the resources under attack, as seen in Figure 3 below, using JSON to serialize the data:

```
{
    "https1": [
        "192.0.2.1:443",
        "198.51.100.2:443",
    ],
    "proxies": [
        "203.0.113.3:3128",
        "[2001:DB8:AC10::1]:3128"
    ],
    "api_urls": "https://apiserver.example.com/api/v1",
}
```

Figure 3: Protected resource identifiers

- Black-list management, which enables a DOTS client to inform the DOTS server about sources to suppress.

- White-list management, which enables a DOTS client to inform the DOTS server about sources from which traffic should always be accepted.

- Filter management, which enables a DOTS client to install or remove traffic filters dropping or rate-limiting unwanted traffic.

- DOTS client provisioning.

Note that while it is possible to exchange the above information before, during or after a DDoS attack, DOTS requires reliable delivery of the this information and does not provide any special means for ensuring timely delivery of it during an attack. In practice, this means that DOTS deployments SHOULD NOT rely on such information being exchanged during a DDoS attack.

2.1. DOTS Operations

The scope of DOTS is focused on the signaling and data exchange between the DOTS client and DOTS server. DOTS does not prescribe any specific deployment models, however DOTS is designed with some specific requirements around the different DOTS agents and their relationships.

First of all, a DOTS agent belongs to an domain, and that domain has an identity which can be authenticated and authorized. DOTS agents communicate with each other over a mutually authenticated signal channel and data channel. However, before they can do so, a service relationship needs to be established between them. The details and means by which this is done is outside the scope of DOTS, however an example would be for an enterprise A (DOTS client) to sign up for DDoS service from provider B (DOTS server). This would establish a (service) relationship between the two that enables enterprise A’s DOTS client to establish a signal channel with provider B’s DOTS server. A and B will authenticate each other, and B can verify that A is authorized for its service.

From an operational and design point of view, DOTS assumes that the above relationship is established prior to a request for DDoS attack mitigation. In particular, it is assumed that bi-directional communication is possible at this time between the DOTS client and DOTS server. Furthermore, it is assumed that additional service provisioning, configuration and information exchange can be performed by use of the data channel, if operationally required. It is not until this point that the mitigation service is available for use.

Once the mutually authenticated signal channel has been established, it will remain in place. This is done to increase the likelihood that the DOTS client can signal the DOTS server for help when the attack target is being flooded, and similarly raise the probability that DOTS server signals reach the client regardless of inbound link congestion. 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.

DDoS mitigation service with the help of an upstream mitigator will often involve some form of traffic redirection whereby traffic destined for the attack target is diverted towards the mitigator, e.g. by use of BGP [RFC4271] or DNS [RFC1034]. The mitigator in turn inspects and scrubs the traffic, and forwards the resulting (hopefully non-attack) traffic to the attack target. Thus, when a

DOTS server receives an attack mitigation request from a DOTS client, it can be viewed as a way of causing traffic redirection for the attack target indicated.

DOTS relies on mutual authentication and the pre-established service relationship between the DOTS client’s domain and the DOTS server’s domain to provide basic authorization. The DOTS server SHOULD enforce additional authorization mechanisms to restrict the mitigation scope a DOTS client can request, but such authorization mechanisms are deployment-specific.

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. Nothing in this document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help coordinating attack response originate. The requests may be in response to an active, ongoing attack against a target in the DOTS client’s domain, but no active attack is required for a DOTS client to request help. Local operators may wish to have upstream mitigators in the network path for an indefinite period, and are restricted only by business relationships when it comes to duration and scope of requested mitigation.

The DOTS client requests attack response coordination from a DOTS server over the signal channel, including in the request the DOTS client’s desired mitigation scoping, as described in [I-D.ietf-dots-requirements]. The actual mitigation scope and countermeasures used in response to the attack are up to the DOTS server and Mitigator operators, as the DOTS client may have a narrow perspective on the ongoing attack. As such, the DOTS client’s request for mitigation should be considered advisory: guarantees of DOTS server availability or mitigation capacity constitute service level agreements and are out of scope for this document.

The DOTS client adjusts mitigation scope and provides available attack details at the direction of its local operator. Such direction may involve manual or automated adjustments in response to feedback from the DOTS server.

To provide a metric of signal health and distinguish an idle signaling session from a disconnected or defunct session, the DOTS client sends a heartbeat over the signal channel to maintain its half of the signaling session. The DOTS client similarly expects a heartbeat from the DOTS server, and MAY consider a signaling session terminated in the extended absence of a DOTS server heartbeat.

2.2.2. DOTS Server

A DOTS server is a DOTS agent capable of receiving, processing and possibly acting on requests for help coordinating attack response from one or more DOTS clients. The DOTS server authenticates and authorizes DOTS clients as described in Signaling Sessions below, and maintains signaling session state, tracking requests for mitigation, reporting on the status of active mitigations, and terminating signaling sessions in the extended absence of a client heartbeat or when a session times out.

Assuming the preconditions discussed below exist, a DOTS client maintaining an active signaling session with a DOTS server may reasonably expect some level of mitigation in response to a request for coordinated attack response.

The DOTS server enforces authorization of DOTS clients’ signals for mitigation. The mechanism of enforcement is not in scope for this document, but is expected to restrict requested mitigation scope to addresses, prefixes, and/or services owned by the DOTS client’s administrative domain, such that a DOTS client from one domain is not able to influence the network path to another domain. A DOTS server MUST reject requests for mitigation of resources not owned by the requesting DOTS client’s administrative domain. A DOTS server MAY also refuse a DOTS client’s mitigation request for arbitrary reasons, within any limits imposed by business or service level agreements between client and server domains. If a DOTS server refuses a DOTS client’s request for mitigation, the DOTS server SHOULD include the refusal reason in the server signal sent to the client.

A DOTS server is in regular contact with one or more mitigators. If a DOTS server accepts a DOTS client’s request for help, the DOTS server forwards a translated form of that request to the mitigator or mitigators responsible for scrubbing attack traffic. Note that the form of the translated request passed from the DOTS server to the mitigator is not in scope: it may be as simple as an alert to mitigator operators, or highly automated using vendor or open application programming interfaces supported by the mitigator. The DOTS server MUST report the actual scope of any mitigation enabled on behalf of a client.

The DOTS server SHOULD retrieve available metrics for any mitigations activated on behalf of a DOTS client, and SHOULD include them in server signals sent to the DOTS client originating the request for mitigation.

To provide a metric of signal health and distinguish an idle signaling session from a disconnected or defunct session, the DOTS server sends a heartbeat over the signal channel to maintain its half of the signaling session. The DOTS server similarly expects a heartbeat from the DOTS client, and MAY consider a signaling session terminated in the extended absence of a DOTS client heartbeat.

2.2.3. DOTS Gateway

Traditional client to server relationships may be expanded by chaining DOTS sessions. This chaining is enabled through "logical concatenation" [RFC7092] of a DOTS server and a DOTS client, resulting in an application analogous to the SIP logical entity of a Back-to-Back User Agent (B2BUA) [RFC3261]. The term DOTS gateway will be used here and the following text will describe some interactions in relation to this application.

A DOTS gateway may be deployed client-side, server-side or both. The gateway may terminate multiple discrete client connections and may aggregate these into a single or multiple DOTS signaling sessions.

The DOTS gateway will appear as a server to its downstream agents and as a client to its upstream agents, a functional concatenation of the DOTS client and server roles, as depicted in Figure 4:

```
                         +-------------+
                         |    | D |    |
         +----+          |    | O |    |         +----+
         | c1 |----------| s1 | T | c2 |---------| s2 |
         +----+          |    | S |    |         +----+
                         |    | G |    |
                         +-------------+
```

Figure 4: DOTS gateway

The DOTS gateway performs full stack DOTS session termination and reorigination between its client and server side. The details of how this is achieved are implementation specific. The DOTS protocol does not include any special features related to DOTS gateways, and hence from a DOTS perspective, whenever a DOTS gateway is present, the DOTS session simply terminates/originates there.

2.3. DOTS Agent Relationships

So far, we have only considered a relatively simple scenario of a single DOTS client associated with a single DOTS server, however DOTS supports more advanced relationships.

A DOTS server may be associated with one or more DOTS clients, and those DOTS clients may belong to different domains. An example scenario is a mitigation provider serving multiple attack targets (Figure 5):

```
      DOTS Clients       DOTS Server
      +---+
      | c |-----------
      +---+           \
      example.org      \
                        \
      +---+              \ +---+
      | c |----------------| S |
      +---+              / +---+
      example.com       /
                       /
      +---+           /
      | c |-----------
      +---+
      example.com        example.net
```

Figure 5: DOTS server with multiple clients

A DOTS client may be associated with one or more DOTS servers, and those DOTS servers may belong to different domains. This may be to ensure high availability or co-ordinate mitigation with more than one directly connected ISP. An example scenario is for an enterprise to have DDoS mitigation service from multiple providers, as shown in Figure 6 below. Operational considerations relating to co-ordinating multiple provider responses are beyond the scope of DOTS.

[[EDITOR’S NOTE: we request working group feedback and discussion of operational considerations relating to coordinating multiple provider responses to a mitigation request.]]

```
      DOTS Client        DOTS Servers
                          +---+
              ------------| S |
             /            +---+
                          dots1.example.net
            /
      +---+/              +---+
      | c |---------------| S |
      +---+\              +---+
                          dots.example.org
            \
             \            +---+
              ------------| S |
                          +---+
      example.com         dots2.example.net
```

Figure 6: Multi-Homed DOTS Client

2.3.1. Gatewayed signaling

As discussed above in Section 2.2.3, a DOTS gateway is a logical function chaining signaling sessions through concatenation of a DOTS server and DOTS client.

An example scenario, as shown in Figure 7 and Figure 8 below, is for an enterprise to have deployed multiple DOTS capable devices which are able to signal intra-domain using TCP [RFC0793] on un-congested links to a DOTS gateway which may then transform these to a UDP [RFC0768] transport inter-domain where connection oriented transports may degrade; this applies to the signal channel only, as the data channel requires a connection-oriented transport. The relationship between the gateway and its upstream agents is opaque to the initial clients.

```
         +---+
         | c |\
         +---+ \              +---+
                \-----TCP-----| D |              +---+
         +---+                | O |              |   |
         | c |--------TCP-----| T |------UDP------| S |
         +---+                | S |              |   |
                /-----TCP-----| G |              +---+
         +---+ /              +---+
         | c |/
         +---+
         example.com       example.com           example.net
         DOTS Clients      DOTS Gateway (DOTSG)  DOTS Server
```

Figure 7: Client-Side Gateway with Aggregation

```
         +---+
         | c |\
         +---+ \              +---+
                \-----TCP-----| D |------UDP------+---+
         +---+                | O |              |   |
         | c |--------TCP-----| T |------UDP------| S |
         +---+                | S |              |   |
                /-----TCP-----| G |------UDP------+---+
         +---+ /              +---+
         | c |/
         +---+
         example.com       example.com           example.net
         DOTS Clients      DOTS Gateway (DOTSG)  DOTS Server
```

Figure 8: Client-Side Gateway without Aggregation

This may similarly be deployed in the inverse scenario where the gateway resides in the server-side domain and may be used to terminate and/or aggregate multiple clients to single transport as shown in figures Figure 9 and Figure 10 below.

```
         +---+
         | c |\
         +---+ \              +---+
                \-----UDP-----| D |               +---+
         +---+                | O |               |   |
         | c |--------TCP-----| T |------TCP------| S |
         +---+                | S |               |   |
                /-----TCP-----| G |               +---+
         +---+ /              +---+
         | c |/
         +---+
         example.com       example.net           example.net
         DOTS Clients      DOTS Gateway (DOTSG)  DOTS Server
```

Figure 9: Server-Side Gateway with Aggregation

```
         +---+
         | c |\
         +---+ \              +---+
                \-----UDP-----| D |------TCP------+---+
         +---+                | O |               |   |
         | c |--------TCP-----| T |------TCP------| S |
         +---+                | S |               |   |
                /-----UDP-----| G |------TCP------+---+
         +---+ /              +---+
         | c |/
         +---+
         example.com       example.net           example.net
         DOTS Clients      DOTS Gateway (DOTSG)  DOTS Server
```

Figure 10: Server-Side Gateway without Aggregation

3. Concepts

3.1. Signaling Sessions

In order for DOTS to be effective as a vehicle for DDoS mitigation requests, one or more DOTS clients must establish ongoing communication with one or more DOTS servers. While the preconditions for enabling DOTS in or among network domains may also involve business relationships, service level agreements, or other formal or informal understandings between network operators, such considerations are out of scope for this document.

An established communication layer between DOTS agents is a Signaling Session. At its most basic, for a DOTS signaling session to exist both signal channel and data channel must be functioning between DOTS agents. That is, under nominal network conditions, signals actively sent from a DOTS client are received by the specific DOTS server intended by the client, and vice versa.

3.1.1. Preconditions

Prior to establishing a signaling session between agents, the owners of the networks, domains, services or applications involved are assumed to have agreed upon the terms of the relationship involved. Such agreements are out of scope for this document, but must be in place for a functional DOTS architecture.

It is assumed that as part of any DOTS service agreement, the DOTS client is provided with all data and metadata required to establish communication with the DOTS server. Such data and metadata would include any cryptographic information necessary to meet the message confidentiality, integrity and authenticity requirement in [I-D.ietf-dots-requirements], and might also include the pool of DOTS server addresses and ports the DOTS client should use for signal and data channel messaging.

3.1.2. Establishing the Signaling Session

With the required business or service agreements in place, the DOTS client initiates a signal session by contacting the DOTS server over the signal channel and the data channel. To allow for DOTS service flexibility, neither the order of contact nor the time interval between channel creations is specified. A DOTS client MAY establish signal channel first, and then data channel, or vice versa.

The methods by which a DOTS client receives the address and associated service details of the DOTS server are not prescribed by this document. For example, a DOTS client may be directly configured to use a specific DOTS server address and port, and directly provided with any data necessary to satisfy the Peer Mutual Authentication requirement in [I-D.ietf-dots-requirements], such as symmetric or asymmetric keys, usernames and passwords, etc. All configuration and authentication information in this scenario is provided out-of-band by the domain operating the DOTS server.

At the other extreme, the architecture in this document allows for a form of DOTS client auto-provisioning. For example, the domain operating the DOTS server or servers might provide the client domain only with symmetric or asymmetric keys to authenticate the provisioned DOTS clients. Only the keys would then be directly configured on DOTS clients, but the remaining configuration required to provision the DOTS clients could be learned through mechanisms similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

The DOTS client SHOULD successfully authenticate and exchange messages with the DOTS server over both signal and data channel as soon as possible to confirm that both channels are operational.

As described in [I-D.ietf-dots-requirements], the DOTS client can configure preferred values for acceptable signal loss, mitigation lifetime, and heartbeat intervals when establishing the signaling session. A signaling session is not active until DOTS agents have agreed on the values for these signaling session parameters, a process defined by the protocol.

Once the DOTS client begins receiving DOTS server signals, the signaling session is active. At any time during the signaling session, the DOTS client MAY use the data channel to adjust initial configuration, manage black- and white-listed prefixes or addresses, leverage vendor-specific extensions, and so on. Note that unlike the signal channel, there is no requirement that the data channel remain operational in attack conditions (See Data Channel Requirements, [I-D.ietf-dots-requirements]).

3.1.3. Maintaining the Signaling Session

DOTS clients and servers periodically send heartbeats to each other over the signal channel, per Operational Requirements discussed in [I-D.ietf-dots-requirements]. DOTS agent operators SHOULD configure the heartbeat interval such that the frequency does not lead to accidental denials of service due to the overwhelming number of heartbeats a DOTS agent must field.

Either DOTS agent may consider a signaling session terminated in the extended absence of a heartbeat from its peer agent. The period of that absence will be established in the protocol definition.

3.2. Modes of Signaling

This section examines the modes of signaling between agents in a DOTS architecture.

3.2.1. Direct Signaling

A signaling session may take the form of direct signaling between the DOTS clients and servers, as shown in Figure 11 below:

```
        +-------------+                            +-------------+
        | DOTS client |<------signal session------>| DOTS server |
        +-------------+                            +-------------+
```

Figure 11: Direct Signaling

In a direct signaling session, DOTS client and server are communicating directly. A direct signaling session MAY exist inter- or intra-domain. The signaling session is abstracted from the underlying networks or network elements the signals traverse: in a direct signaling session, the DOTS client and server are logically peer DOTS agents.

3.2.2. Redirected Signaling

In certain circumstances, a DOTS server may want to redirect a DOTS client to an alternative DOTS server for a signaling session. Such circumstances include but are not limited to:

- Maximum number of signaling sessions with clients has been reached;

- Mitigation capacity exhaustion in the Mitigator with which the specific DOTS server is communicating;

- Mitigator outage or other downtime, such as scheduled maintenance;

- Scheduled DOTS server maintenance;

- Scheduled modifications to the network path between DOTS server and DOTS client.

A basic redirected signaling session resembles the following, as shown in Figure 12:

```
           +-------------+                            +---------------+
           |             |<-(1)-- signal session 1 -->|               |
           |             |                            |               |
           |             |<=(2)== redirect to B ======|               |
           | DOTS client |                            | DOTS server A |
           |             |X-(4)-- signal session 1 --X|               |
           |             |                            |               |
           |             |                            |               |
           +-------------+                            +---------------+
                  ^
                  |
                 (3) signal session 2
                  |
                  v
           +---------------+
           | DOTS server B |
           +---------------+
```

Figure 12: Redirected Signaling

1. Previously established signaling session 1 exists between a DOTS client and DOTS server with address A.

2. DOTS server A sends a server signal redirecting the client to DOTS server B.

3. If the DOTS client does not already have a separate signaling session with the redirection target, the DOTS client initiates and establishes a signaling session with DOTS server B as described above.

4. Having redirected the DOTS client, DOTS server A ceases sending server signals. The DOTS client likewise stops sending client signals to DOTS server A. Signal session 1 is terminated.

[[EDITOR’S NOTE: we request working group feedback and discussion of the need for redirected signaling.]]

3.2.3. Recursive Signaling

DOTS is centered around improving the speed and efficiency of coordinated response to DDoS attacks. One scenario not yet discussed involves coordination among federated domains operating DOTS servers and mitigators.

In the course of normal DOTS operations, a DOTS client communicates the need for mitigation to a DOTS server, and that server initiates mitigation on a mitigator with which the server has an established service relationship. The operator of the mitigator may in turn monitor mitigation performance and capacity, as the attack being mitigated may grow in severity beyond the mitigating domain’s capabilities.

The operator of the mitigator has limited options in the event a DOTS client-requested mitigation is being overwhelmed by the severity of the attack. Out-of-scope business or service level agreements may permit the mitigating domain to drop the mitigation and let attack traffic flow unchecked to the target, but this is only encourages attack escalation. In the case where the mitigating domain is the upstream service provider for the attack target, this may mean the mitigating domain and its other services and users continue to suffer the incidental effects of the attack.

A recursive signaling model as shown in Figure 13 below offers an alternative. In a variation of the primary use case "Successful Automatic or Operator-Assisted CPE or PE Mitigators Request Upstream DDoS Mitigation Services" described in [I-D.ietf-dots-use-cases], an domain operating a DOTS server and mitigation also operates a DOTS client. This DOTS client has an established signaling session with a DOTS server belonging to a separate administrative domain.

With these preconditions in place, the operator of the mitigator being overwhelmed or otherwise performing inadequately may request mitigation for the attack target from this separate DOTS-aware domain. Such a request recurses the originating mitigation request to the secondary DOTS server, in the hope of building a cumulative mitigation against the attack:

```
                        example.net domain
                        . . . . . . . . . . . . . . . . .
                        .    Gn                         .
          +----+    A   .  +----+       +-----------+   .
          | Cc |<--------->| Sn |~~~~~~~| Mitigator |   .
          +----+        .  +====+       |     Mn    |   .
                        .  | Cn |       +-----------+   .
        example.com     .  +----+                       .
           client       .    ^                          .
                        . . .|. . . . . . . . . . . . . .
                             |
                           B |
                             |
                        . . .|. . . . . . . . . . . . . .
                        .    v                          .
                        .  +----+       +-----------+   .
                        .  | So |~~~~~~~| Mitigator |   .
                        .  +----+       |     Mo    |   .
                        .               +-----------+   .
                        .                               .
                        . . . . . . . . . . . . . . . . .
                        example.org domain
```

Figure 13: Recursive Signaling

In Figure 13 above, client Cc signals a request for mitigation across inter-domain signaling session A to the DOTS server Sn belonging to the example.net domain. DOTS server Sn enables mitigation on mitigator Mn. DOTS server Sn is half of DOTS gateway Gn, being deployed logically back-to-back with DOTS client Cn, which has pre-existing inter-domain signaling session B with the DOTS server So belonging to the example.org domain. At any point, DOTS server Sn MAY recurse an on-going mitigation request through DOTS client Cn to DOTS server So, in the expectation that mitigator Mo will be activated to aid in the defense of the attack target.

Recursive signaling is opaque to the DOTS client. To maximize mitigation visibility to the DOTS client, however, the recursing domain SHOULD provide recursed mitigation feedback in signals reporting on mitigation status to the DOTS client. For example, the recursing domain’s mitigator should incorporate into mitigation status messages available metrics such as dropped packet or byte counts from the recursed mitigation.

DOTS clients involved in recursive signaling MUST be able to withdraw requests for mitigation without warning or justification, per [I-D.ietf-dots-requirements].

Operators recursing mitigation requests MAY maintain the recursed mitigation for a brief, protocol-defined period in the event the DOTS client originating the mitigation withdraws its request for help, as per the discussion of managing mitigation toggling in the operational requirements ([I-D.ietf-dots-requirements]). Service or business agreements between recursing domains are not in scope for this document.

[[EDITOR’S NOTE: Recursive signaling raises questions about operational and data privacy, as well as what level of visibility a client has into the recursed mitigation. We ask the working group for feedback and additional discussion of these issues to help settle the way forward.]]

3.2.4. Anycast Signaling

The DOTS architecture does not assume the availability of anycast within a DOTS deployment, but neither does the architecture exclude it. Domains operating DOTS servers MAY deploy DOTS servers with an anycast Service Address as described in BCP 126 [RFC4786]. In such a deployment, DOTS clients connecting to the DOTS Service Address may be communicating with distinct DOTS servers, depending on the network configuration at the time the DOTS clients connect. Among other benefits, anycasted signaling potentially offers the following:

- Simplified DOTS client configuration, including service discovery through the methods described in [RFC7094]. In this scenario, the "instance discovery" message would be a DOTS client initiating a signaling session to the DOTS server anycast Service Address, to which the DOTS server would reply with a redirection to the DOTS server unicast address the client should use for DOTS.

- Region- or customer-specific deployments, in which the DOTS Service Addresses route to distinct DOTS servers depending on the client region or the customer network in which a DOTS client resides.

- Operational resiliency, spreading DOTS signaling traffic across the DOTS server domain’s networks, and thereby also reducing the potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

As long as network configuration remains stable, anycast DOTS signaling is to the individual DOTS client indistinct from direct signaling. However, the operational challenges inherent in anycast signaling are anything but negligible, and DOTS server operators must carefully weigh the risks against the benefits before deploying.

While the DOTS signal channel primarily operates over UDP per [I-D.ietf-dots-requirements], the signal channel also requires mutual authentication between DOTS agents, with associated security state on both ends. The resulting considerations therefore superficially resemble the deployment of anycast DNS over DTLS, as described in [I-D.ietf-dprive-dnsodtls], but the similiarities only go so far.

Network instability is of particular concern with anycast signaling, as DOTS signaling sessions are expected to be long-lived, and potentially operating under congested network conditions caused by a volumetric DDoS attack.

For example, a network configuration altering the route to the DOTS server during active anycast signaling may cause the DOTS client to send messages to a DOTS server other than the one with which it initially established a signaling session. That second DOTS server may not have the security state of the existing session, forcing the DOTS client to initialize a new signaling session. This challenge may in part be mitigated by use of pre-shared keys, as described in [I-D.ietf-tls-tls13], but keying material must be available to all DOTS servers sharing the anycast Service Address in that case.

While the DOTS client will try to establish a new signaling session with the DOTS server now acting as the anycast DOTS Service Address, the link between DOTS client and server may be congested with attack traffic, making signal session establishment difficult. In such a scenario, anycast Service Address instability becomes a sort of signal session flapping, with obvious negative consequences for the DOTS deployment.

Anycast signaling deployments similarly must also take into account active mitigations. Active mitigations initiated through a signaling session may involve diverting traffic to a scrubbing center. If the signaling session flaps due to anycast changes as described above, mitigation may also flap as the DOTS servers sharing the anycast DOTS service address toggles mitigation on detecting signaling session loss, depending on whether the client has configured mitigation on loss of signal.

[[EDITOR’S NOTE: We request feedback from the working group regarding the complexities inherent in an anycast DOTS deployment. Outside of using anycast for service discovery, significant challenges need to be overcome, particularly when dealing with security and mitigation state, and the resulting operational complexity may outweigh the expected benefits.]]

3.3. Triggering Requests for Mitigation

[I-D.ietf-dots-requirements] places no limitation on the circumstances in which a DOTS client operator may request mitigation, nor does it demand justification for any mitigation request, thereby reserving operational control over DDoS defense for the domain requesting mitigation. This architecture likewise does not prescribe the network conditions and mechanisms triggering a mitigation request from a DOTS client.

However, considering selected possible mitigation triggers from an architectural perspective offers a model for alternative or unanticipated triggers for DOTS deployments. In all cases, what network conditions merit a mitigation request are at the discretion of the DOTS client operator.

The interfaces required to trigger the mitigation request in the following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

A DOTS client operator may manually prepare a request for mitigation, including scope and duration, and manually instruct the DOTS client to send the mitigation request to the DOTS server. In context, a manual request is a request directly issued by the operator without automated decision-making performed by a device interacting with the DOTS client. Modes of manual mitigation requests include an operator entering a command into a text interface, or directly interacting with a graphical interface to send the request.

An operator might do this, for example, in response to notice of an attack delivered by attack detection equipment or software, and the alerting detector lacks interfaces or is not configured to use available interfaces to translate the alert to a mitigation request automatically.

In a variation of the above scenario, the operator may have preconfigured on the DOTS client mitigation request for various resources in the operator’s domain. When notified of an attack, the DOTS client operator manually instructs the DOTS client to send the preconfigured mitigation request for the resources under attack.

A further variant involves recursive signaling, as described in Section 3.2.3. The DOTS client in this case is the second half of a DOTS gateway (back-to-back DOTS server and client). As in the previous scenario, the scope and duration of the mitigation request are pre-existing, but in this case are derived from the mitigation request received from a downstream DOTS client by the DOTS server. Assuming the preconditions required by Section 3.2.3 are in place, the DOTS gateway operator may at any time manually request mitigation from an upstream DOTS server, sending a mitigation request derived from the downstream DOTS client’s request.

The motivations for a DOTS client operator to request mitigation manually are not prescribed by this architecture, but are expected to include some of the following:

- Notice of an attack delivered via e-mail or alternative messaging

- Notice of an attack delivered via phone call

- Notice of an attack delivered through the interface(s) of networking monitoring software deployed in the operator’s domain

- Manual monitoring of network behavior through network monitoring software

3.3.2. Automated Conditional Mitigation Request

Unlike manual mitigation requests, which depend entirely on the DOTS client operator’s capacity to react with speed and accuracy to every detected or detectable attack, mitigation requests triggered by detected attack conditions reduce the operational burden on the DOTS client operator, and minimize the latency between attack detection and the start of mitigation.

Mitigation requests are triggered in this scenario by operator-specified network conditions. Attack detection is deployment-specific, and not constrained by this architecture. Similarly the specifics of a condition are left to the discretion of the operator, though common conditions meriting mitigation include the following:

- Detected attack exceeding a rate in packets per second (pps).

- Detected attack exceeding a rate in bytes per second (bps).

- Detected resource exhaustion in an attack target.

- Detected resource exhaustion in the local domain’s mitigator.

- Number of open connections to an attack target.

- Number of attack sources in a given attack.

- Number of active attacks against targets in the operator’s domain.

- Conditional detection developed through arbitrary statistical analysis or deep learning techniques.

- Any combination of the above.

When automated conditional mitigation requests are enabled, violations of any of the above conditions, or any additional operator-defined conditions, will trigger a mitigation request from the DOTS client to the DOTS server. The interfaces between the application detecting the condition violation and the DOTS client are implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

To maintain a signaling session, the DOTS client and the DOTS server exchange regular but infrequent messages across the signaling channel. In the absence of an attack, the probability of message loss in the signaling channel should be extremely low. Under attack conditions, however, some signal loss may be anticipated as attack traffic congests the link, depending on the attack type.

While [I-D.ietf-dots-requirements] specifies the DOTS protocol be robust when signaling under attack conditions, there are nevertheless scenarios in which the DOTS signal is lost in spite of protocol best efforts. To handle such scenarios, a DOTS client operator may configure the signaling session to trigger mitigation when the DOTS server ceases receiving DOTS client signals (or vice versa) beyond the miss count or period permitted by the protocol.

The impact of mitigating due to loss of signal in either direction must be considered carefully before enabling it. Signal loss is not caused by links congested with attack traffic alone, and as such mitigation requests triggered by signal channel degradation in either direction may incur unnecessary costs, in network performance and operational expense alike.

4. Security Considerations

This section describes identified security considerations for the DOTS architecture.

DOTS is at risk from three primary attack vectors: agent impersonation, traffic injection and signal blocking. These vectors may be exploited individually or in concert by an attacker to confuse, disable, take information from, or otherwise inhibit DOTS agents.

Any attacker with the ability to impersonate a legitimate client or server or, indeed, inject false messages into the stream may potentially trigger/withdraw traffic redirection, trigger/cancel mitigation activities or subvert black/whitelists. From an architectural standpoint, operators SHOULD ensure best current practices for secure communication are observed for data and signal channel confidentiality, integrity and authenticity. Care must be taken to ensure transmission is protected by appropriately secure means, reducing attack surface by exposing only the minimal required services or interfaces. Similarly, received data at rest SHOULD be stored with a satisfactory degree of security.

As many mitigation systems employ diversion to scrub attack traffic, operators of DOTS agents SHOULD ensure signaling sessions are resistant to Man-in-the-Middle (MitM) attacks. An attacker with control of a DOTS client may negatively influence network traffic by requesting and withdrawing requests for mitigation for particular prefixes, leading to route or DNS flapping.

Any attack targeting the availability of DOTS servers may disrupt the ability of the system to receive and process DOTS signals resulting in failure to fulfill a mitigation request. DOTS agents SHOULD be given adequate protections, again in accordance with best current practices for network and host security.

5. Acknowledgments

Thanks to Matt Richardson and Med Boucadair for their comments and suggestions.

6. Change Log

2016-03-18 Initial revision

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-02 (work in progress), July 2016.

[I-D.ietf-dots-use-cases] Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., and L. Xia, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-01 (work in progress), March 2016.

[I-D.ietf-dprive-dnsodtls] Reddy, T., Wing, D., and P. Patil, "Specification for DNS over Datagram Transport Layer Security (DTLS)", draft-ietf-dprive-dnsodtls-12 (work in progress), September 2016.

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-18 (work in progress), October 2016.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[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, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <http://www.rfc-editor.org/info/rfc4732>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <http://www.rfc-editor.org/info/rfc4786>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://www.rfc-editor.org/info/rfc6763>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <http://www.rfc-editor.org/info/rfc7092>.

[RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <http://www.rfc-editor.org/info/rfc7094>.

Authors’ Addresses

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

EMail: amortensen@arbor.net

Flemming Andreasen
Cisco Systems, Inc.
United States

EMail: fandreas@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

EMail: tireddy@cisco.com

Christopher Gray
Comcast, Inc.
United States

EMail: Christopher_Gray3@cable.comcast.com

Rich Compton
Charter Communications, Inc.

EMail: Rich.Compton@charter.com

Nik Teague
Verisign, Inc.
United States

EMail: nteague@verisign.com

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

A. Mortensen, Ed.
Forcepoint
T. Reddy, Ed.
McAfee, Inc.
F. Andreasen
Cisco
N. Teague
Iron Mountain
R. Compton
Charter

March 6, 2020

Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
draft-ietf-dots-architecture-18

Abstract

This document describes an architecture for establishing and maintaining Distributed Denial of Service (DDoS) Open Threat Signaling (DOTS) within and between domains. The document does not specify protocols or protocol extensions, instead focusing on defining architectural relationships, components and concepts used in a DOTS deployment.

Status of This Memo

Copyright Notice

Table of Contents

1. Context and Motivation

Signaling the need for help to defend against an active distributed denial of service (DDoS) attack requires a common understanding of mechanisms and roles among the parties coordinating defensive response. The signaling layer and supplementary messaging is the focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of coordinating defensive measures among willing peers to mitigate attacks quickly and efficiently, enabling hybrid attack responses coordinated locally at or near the target of an active attack, or anywhere in-path between attack sources and target. Sample DOTS use cases are elaborated in [I-D.ietf-dots-use-cases].

This document describes an architecture used in establishing, maintaining or terminating a DOTS relationship within a domain or between domains.

1.1. Terminology

1.1.1. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.1.2. Definition of Terms

This document uses the terms defined in [RFC8612].

1.2. Scope

In this architecture, DOTS clients and servers communicate using DOTS signal channel [I-D.ietf-dots-signal-channel] and data channel [I-D.ietf-dots-data-channel] protocols.

The DOTS architecture presented here is applicable across network administrative domains - for example, between an enterprise domain and the domain of a third-party attack mitigation service - as well as to a single administrative domain. DOTS is generally assumed to be most effective when aiding coordination of attack response between two or more participating networks, but single domain scenarios are valuable in their own right, as when aggregating intra-domain DOTS client signals for inter-domain coordinated attack response.

This document does not address any administrative or business agreements that may be established between involved DOTS parties. Those considerations are out of scope. Regardless, this document assumes necessary authentication and authorization mechanisms are put in place so that only authorized clients can invoke the DOTS service.

A detailed set of DOTS requirements are discussed in [RFC8612], and the DOTS architecture is designed to follow those requirements. Only new behavioral requirements are described in this document.

1.3. Assumptions

This document makes the following assumptions:

- All domains in which DOTS is deployed are assumed to offer the required connectivity between DOTS agents and any intermediary network elements, but the architecture imposes no additional limitations on the form of connectivity.

- Congestion and resource exhaustion are intended outcomes of a DDoS attack [RFC4732]. Some operators may utilize non-impacted paths or networks for DOTS, but in general conditions should be assumed to be hostile and DOTS must be able to function in all circumstances, including when the signaling path is significantly impaired. Congestion control requirements are discussed in Section 3 of [RFC8612]. DOTS signal channel defined in [I-D.ietf-dots-signal-channel] is designed to be extremely resilient under extremely hostile network conditions and provides continued contact between DOTS agents even as DDoS attack traffic saturates the link.

- There is no universal DDoS attack scale threshold triggering a coordinated response across administrative domains. A network domain administrator, or service or application owner may arbitrarily set attack scale threshold triggers, or manually send requests for mitigation.

- Mitigation requests may be sent to one or more upstream DOTS servers based on criteria determined by DOTS client administrators and the underlying network configuration. The number of DOTS servers with which a given DOTS client has established communications is determined by local policy and is deployment-specific. For example, a DOTS client of a multi-homed network may support built-in policies to establish DOTS relationships with DOTS servers located upstream of each interconnection link.

- The mitigation capacity and/or capability of domains receiving requests for coordinated attack response is opaque to the domains sending the request. The domain receiving the DOTS client signal may or may not have sufficient capacity or capability to filter any or all DDoS attack traffic directed at a target. In either case, the upstream DOTS server may redirect a request to another DOTS server. Redirection may be local to the redirecting DOTS server's domain, or may involve a third-party domain.

- DOTS client and server signals, as well as messages sent through the data channel, are sent across any transit networks with the same probability of delivery as any other traffic between the DOTS client domain and the DOTS server domain. Any encapsulation required for successful delivery is left untouched by transit network elements. DOTS server and DOTS client cannot assume any preferential treatment of DOTS signals. Such preferential treatment may be available in some deployments (e.g., intra-domain scenarios), and the DOTS architecture does not preclude its use when available. However, DOTS itself does not address how that may be done.

- The architecture allows for, but does not assume, the presence of Quality of Service (QoS) policy agreements between DOTS-enabled peer networks or local QoS prioritization aimed at ensuring delivery of DOTS messages between DOTS agents. QoS is an operational consideration only, not a functional part of the DOTS architecture.

- The signal and data channels are loosely coupled, and might not terminate on the same DOTS server. How the DOTS servers synchronize the DOTS configuration is out of scope of this specification.

2. DOTS Architecture

The basic high-level DOTS architecture is illustrated in Figure 1:

```
    +-----------+            +-------------+
    | Mitigator | ~~~~~~~~~~ | DOTS Server |
    +-----------+            +-------------+
                                    |
                                    |
                                    |
    +---------------+        +-------------+
    | Attack Target | ~~~~~~ | DOTS Client |
    +---------------+        +-------------+
```

Figure 1: Basic DOTS Architecture

A simple example instantiation of the DOTS architecture could be an enterprise as the attack target for a volumetric DDoS attack, and an upstream DDoS mitigation service as the mitigator. The service provided by the mitigator is called: DDoS mitigation service. The enterprise (attack target) is connected to the Internet via a link that is getting saturated, and the enterprise suspects it is under DDoS attack. The enterprise has a DOTS client, which obtains information about the DDoS attack, and signals the DOTS server for help in mitigating the attack. The DOTS server in turn invokes one or more mitigators, which are tasked with mitigating the actual DDoS attack, and hence aim to suppress the attack traffic while allowing valid traffic to reach the attack target.

The scope of the DOTS specifications is the interfaces between the DOTS client and DOTS server. The interfaces to the attack target and the mitigator are out of scope of DOTS. Similarly, the operation of both the attack target and the mitigator is out of scope of DOTS. Thus, DOTS specifies neither how an attack target decides it is under DDoS attack, nor does DOTS specify how a mitigator may actually mitigate such an attack. A DOTS client's request for mitigation is advisory in nature, and might not lead to any mitigation at all, depending on the DOTS server domain's capacity and willingness to mitigate on behalf of the DOTS client domain.

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 sessions by connecting to the provided DOTS server addresses.

As illustrated in Figure 2, there are two interfaces between a DOTS server and a DOTS client; a signal channel and (optionally) a data channel.

```
+---------------+                                 +---------------+
|               | <------- Signal Channel ------> |               |
|  DOTS Client  |                                 |  DOTS Server  |
|               | <=======  Data Channel  ======> |               |
+---------------+                                 +---------------+
```

Figure 2: DOTS Interfaces

The primary purpose of the signal channel is for a DOTS client to ask a DOTS server for help in mitigating an attack, and for the DOTS server to inform the DOTS client about the status of such mitigation. The DOTS client does this by sending a client signal, which contains information about the attack target(s). The client signal may also include telemetry information about the attack, if the DOTS client has such information available. The DOTS server in turn sends a server signal to inform the DOTS client of whether it will honor the mitigation request. Assuming it will, the DOTS server initiates attack mitigation, and periodically informs the DOTS client about the status of the mitigation. Similarly, the DOTS client periodically informs the DOTS server about the client’s status, which at a minimum provides client (attack target) health information, but it should also include efficacy information about the attack mitigation as it is now seen by the client. At some point, the DOTS client may decide to terminate the server-side attack mitigation, which it indicates to the DOTS server over the signal channel. A mitigation may also be terminated if a DOTS client-specified mitigation lifetime is exceeded. Note that the signal channel may need to operate over a link that is experiencing a DDoS attack and hence is subject to severe packet loss and high latency.

While DOTS is able to request mitigation with just the signal channel, the addition of the DOTS data channel provides for additional and more efficient capabilities. The primary purpose of the data channel is to support DOTS related configuration and policy information exchange between the DOTS client and the DOTS server. Examples of such information include, but are not limited to:

- Creating identifiers, such as names or aliases, for resources for which mitigation may be requested. Such identifiers may then be used in subsequent signal channel exchanges to refer more efficiently to the resources under attack.

- Drop-list management, which enables a DOTS client to inform the DOTS server about sources to suppress.

- Accept-list management, which enables a DOTS client to inform the DOTS server about sources from which traffic is always accepted.

- Filter management, which enables a DOTS client to install or remove traffic filters dropping or rate-limiting unwanted traffic.

- DOTS client provisioning.

Note that while it is possible to exchange the above information before, during or after a DDoS attack, DOTS requires reliable delivery of this information and does not provide any special means for ensuring timely delivery of it during an attack. In practice, this means that DOTS deployments should rely on such information being exchanged only under normal traffic conditions.

2.1. DOTS Operations

DOTS does not prescribe any specific deployment models, however DOTS is designed with some specific requirements around the different DOTS agents and their relationships.

First of all, a DOTS agent belongs to a domain that has an identity which can be authenticated and authorized. DOTS agents communicate with each other over a mutually authenticated signal channel and (optionally) data channel. However, before they can do so, a service relationship needs to be established between them. The details and means by which this is done is outside the scope of DOTS, however an example would be for an enterprise A (DOTS client) to sign up for DDoS service from provider B (DOTS server). This would establish a (service) relationship between the two that enables enterprise A's DOTS client to establish a signal channel with provider B's DOTS server. A and B will authenticate each other, and B can verify that A is authorized for its service.

From an operational and design point of view, DOTS assumes that the above relationship is established prior to a request for DDoS attack mitigation. In particular, it is assumed that bi-directional communication is possible at this time between the DOTS client and DOTS server. Furthermore, it is assumed that additional service provisioning, configuration and information exchange can be performed by use of the data channel, if operationally required. It is not until this point that the mitigation service is available for use.

Once the mutually authenticated signal channel has been established, it will remain active. This is done to increase the likelihood that the DOTS client can signal the DOTS server for help when the attack target is being flooded, and similarly raise the probability that DOTS server signals reach the client regardless of inbound link congestion. 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.

DDoS mitigation with the help of an upstream mitigator may involve some form of traffic redirection whereby traffic destined for the attack target is steered towards the mitigator. Common mechanisms to achieve this redirection depend on BGP [RFC4271] and DNS [RFC1035]. The mitigator in turn inspects and scrubs the traffic, and forwards the resulting (hopefully non-attack) traffic to the attack target. Thus, when a DOTS server receives an attack mitigation request from a DOTS client, it can be viewed as a way of causing traffic redirection for the attack target indicated.

DOTS relies on mutual authentication and the pre-established service relationship between the DOTS client domain and the DOTS server domain to provide authorization. The DOTS server should enforce authorization mechanisms to restrict the mitigation scope a DOTS client can request, but such authorization mechanisms are deployment-specific.

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. Nothing in this document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help coordinating attack response originate. The requests may be in response to an active, ongoing attack against a target in the DOTS client domain, but no active attack is required for a DOTS client to request help. Operators may wish to have upstream mitigators in the network path for an indefinite period, and are restricted only by business relationships when it comes to duration and scope of requested mitigation.

The DOTS client requests attack response coordination from a DOTS server over the signal channel, including in the request the DOTS client's desired mitigation scoping, as described in [RFC8612] (SIG-008). The actual mitigation scope and countermeasures used in response to the attack are up to the DOTS server and mitigator operators, as the DOTS client may have a narrow perspective on the ongoing attack. As such, the DOTS client's request for mitigation should be considered advisory: guarantees of DOTS server availability or mitigation capacity constitute service level agreements and are out of scope for this document.

The DOTS client adjusts mitigation scope and provides available mitigation feedback (e.g., mitigation efficacy) at the direction of its local administrator. Such direction may involve manual or automated adjustments in response to updates from the DOTS server.

To provide a metric of signal health and distinguish an idle signal channel from a disconnected or defunct session, the DOTS client sends a heartbeat over the signal channel to maintain its half of the channel. The DOTS client similarly expects a heartbeat from the DOTS server, and may consider a session terminated in the extended absence of a DOTS server heartbeat.

2.2.2. DOTS Server

A DOTS server is a DOTS agent capable of receiving, processing and possibly acting on requests for help coordinating attack response from DOTS clients. The DOTS server authenticates and authorizes DOTS clients as described in Section 3.1, and maintains session state, tracking requests for mitigation, reporting on the status of active mitigations, and terminating sessions in the extended absence of a client heartbeat or when a session times out.

Assuming the preconditions discussed below exist, a DOTS client maintaining an active session with a DOTS server may reasonably expect some level of mitigation in response to a request for coordinated attack response.

For a given DOTS client (administrative) domain, the DOTS server needs to be able to determine whether a given resource is in that domain. For example, this could take the form of associating a set of IP addresses and/or prefixes per DOTS client domain. The DOTS server enforces authorization of signals for mitigation, filtering rules and aliases for resources from DOTS clients. The mechanism of enforcement is not in scope for this document, but is expected to restrict mitigation requests, filtering rules and aliases scope to addresses, prefixes, and/or services owned by the DOTS client domain, such that a DOTS client from one domain is not able to influence the network path to another domain. A DOTS server MUST reject mitigation requests, filtering rules and aliases for resources not owned by the requesting DOTS client's administrative domain. The exact mechanism for the DOTS servers to validate that the resources are within the scope of the DOTS client domain is deployment-specific. For example, if the DOTS client domain uses Provider-Aggregatable prefixes for its resources and leverages the DDoS mitigation service of the Internet Transit Provider (ITP), the ITP knows the prefixes assigned to the DOTS client domain because they are assigned by the ITP itself. However, if the DDoS Mitigation is offered by a third party DDoS mitigation service provider, it does not know the resources owned by the DOTS client domain. The DDoS mitigation service provider and the DOTS client domain can opt to use the identifier validation challenges discussed in [RFC8555] and [I-D.ietf-acme-ip] to identify whether the DOTS client domain actually controls the resources. The challenges for validating control of resources must be performed when no attack traffic is present and works only for "dns" and "ip" identifier types. Further, if the DOTS client lies about the resources owned by the DOTS client domain, the DDoS mitigation service provider can impose penalties for violating the service level agreement. A DOTS server MAY also refuse a DOTS client's mitigation request for arbitrary reasons, within any limits imposed by business or service level agreements between client and server domains. If a

DOTS server refuses a DOTS client’s request for mitigation, the DOTS server MUST include the refusal reason in the server signal sent to the client.

A DOTS server is in regular contact with one or more mitigators. If a DOTS server accepts a DOTS client’s request for help, the DOTS server forwards a translated form of that request to the mitigator(s) responsible for scrubbing attack traffic. Note that the form of the translated request passed from the DOTS server to the mitigator is not in scope: it may be as simple as an alert to mitigator operators, or highly automated using vendor or open application programming interfaces supported by the mitigator. The DOTS server MUST report the actual scope of any mitigation enabled on behalf of a client.

The DOTS server SHOULD retrieve available metrics for any mitigations activated on behalf of a DOTS client, and SHOULD include them in server signals sent to the DOTS client originating the request for mitigation.

To provide a metric of signal health and distinguish an idle signal channel from a disconnected or defunct channel, the DOTS server MUST send a heartbeat over the signal channel to maintain its half of the channel. The DOTS server similarly expects a heartbeat from the DOTS client, and MAY consider a session terminated in the extended absence of a DOTS client heartbeat.

2.2.3. DOTS Gateway

Traditional client/server relationships may be expanded by chaining DOTS sessions. This chaining is enabled through "logical concatenation" of a DOTS server and a DOTS client, resulting in an application analogous to the Session Initiation Protocol (SIP) [RFC3261] logical entity of a Back-to-Back User Agent (B2BUA) [RFC7092]. The term DOTS gateway is used here in the descriptions of selected scenarios involving this application.

A DOTS gateway may be deployed client-side, server-side or both. The gateway may terminate multiple discrete client connections and may aggregate these into a single or multiple DOTS sessions.

The DOTS gateway will appear as a server to its downstream agents and as a client to its upstream agents, a functional concatenation of the DOTS client and server roles, as depicted in Figure 3:

```
                         +-------------+
                         |    | D |    |
         +----+          |    | O |    |         +----+
         | c1 |----------| s1 | T | c2 |---------| s2 |
         +----+          |    | S |    |         +----+
                         |    | G |    |
                         +-------------+
```

Figure 3: DOTS gateway

The DOTS gateway MUST perform full stack DOTS session termination and reorigination between its client and server side. The details of how this is achieved are implementation specific.

2.3. DOTS Agent Relationships

So far, we have only considered a relatively simple scenario of a single DOTS client associated with a single DOTS server, however DOTS supports more advanced relationships.

A DOTS server may be associated with one or more DOTS clients, and those DOTS clients may belong to different domains. An example scenario is a mitigation provider serving multiple attack targets (Figure 4).

```
      DOTS clients       DOTS server
      +---+
      | c |-----------
      +---+           \
      c1.example.org   \
                        \
      +---+              \ +---+
      | c |----------------| S |
      +---+              / +---+
      c1.example.com    /  dots1.example.net
                       /
      +---+           /
      | c |-----------
      +---+
      c2.example.com
```

Figure 4: DOTS server with multiple clients

A DOTS client may be associated with one or more DOTS servers, and those DOTS servers may belong to different domains. This may be to ensure high availability or co-ordinate mitigation with more than one directly connected ISP. An example scenario is for an enterprise to have DDoS mitigation service from multiple providers, as shown in Figure 5.

```
      DOTS client        DOTS servers
                          +---+
               -----------| S |
              /           +---+
             /            dots1.example.net
            /
      +---+/              +---+
      | c |---------------| S |
      +---+\              +---+
            \             dots.example.org
             \
              \           +---+
               -----------| S |
                          +---+
      c.example.com       dots2.example.net
```

Figure 5: Multi-Homed DOTS Client

Deploying a multi-homed client requires extra care and planning, as the DOTS servers with which the multi-homed client communicates might not be affiliated. Should the multi-homed client simultaneously request for mitigation from all servers with which it has established signal channels, the client may unintentionally inflict additional network disruption on the resources it intends to protect. In one of the worst cases, a multi-homed DOTS client could cause a permanent routing loop of traffic destined for the client's protected services, as the uncoordinated DOTS servers' mitigators all try to divert that traffic to their own scrubbing centers.

The DOTS protocol itself provides no fool-proof method to prevent such self-inflicted harms as a result deploying multi-homed DOTS clients. If DOTS client implementations nevertheless include support for multi-homing, they are expected to be aware of the risks, and consequently to include measures aimed at reducing the likelihood of negative outcomes. Simple measures might include:

- Requesting mitigation serially, ensuring only one mitigation request for a given address space is active at any given time;

- Dividing the protected resources among the DOTS servers, such that no two mitigators will be attempting to divert and scrub the same traffic;

- o Restricting multi-homing to deployments in which all DOTS servers are coordinating management of a shared pool of mitigation resources.

2.3.1. Gatewayed Signaling

As discussed in Section 2.2.3, a DOTS gateway is a logical function chaining DOTS sessions through concatenation of a DOTS server and DOTS client.

An example scenario, as shown in Figure 6 and Figure 7, is for an enterprise to have deployed multiple DOTS capable devices which are able to signal intra-domain using TCP [RFC0793] on un-congested links to a DOTS gateway which may then transform these to a UDP [RFC0768] transport inter-domain where connection oriented transports may degrade; this applies to the signal channel only, as the data channel requires a connection-oriented transport. The relationship between the gateway and its upstream agents is opaque to the initial clients.

```
+---+
| c |\
+---+ \              +---+
       \-----TCP-----| D |               +---+
+---+                | O |               |   |
| c |--------TCP-----| T |------UDP------| S |
+---+                | S |               |   |
       /-----TCP-----| G |               +---+
+---+ /              +---+
| c |/
+---+
example.com       example.com           example.net
DOTS clients      DOTS gateway (DOTSG)  DOTS server
```

Figure 6: Client-Side Gateway with Aggregation

```
      +---+
      | c |\
      +---+ \              +---+
             \-----TCP-----| D |------UDP------+---+
      +---+                | O |               |   |
      | c |--------TCP-----| T |------UDP------| S |
      +---+                | S |               |   |
             /-----TCP-----| G |------UDP------+---+
      +---+ /              +---+
      | c |/
      +---+
      example.com       example.com           example.net
      DOTS clients      DOTS gateway (DOTSG)  DOTS server
```

Figure 7: Client-Side Gateway without Aggregation

This may similarly be deployed in the inverse scenario where the gateway resides in the server-side domain and may be used to terminate and/or aggregate multiple clients to single transport as shown in figures Figure 8 and Figure 9.

```
      +---+
      | c |\
      +---+ \              +---+
             \-----UDP-----| D |               +---+
      +---+                | O |               |   |
      | c |--------TCP-----| T |------TCP------| S |
      +---+                | S |               |   |
             /-----TCP-----| G |               +---+
      +---+ /              +---+
      | c |/
      +---+
      example.com       example.net           example.net
      DOTS clients      DOTS gateway (DOTSG)  DOTS server
```

Figure 8: Server-Side Gateway with Aggregation

```
         +---+
         | c |\
         +---+ \              +---+
                \-------------| D |------TCP------+---+
         +---+                | O |               |   |
         | c |--------TCP-----| T |------TCP------| S |
         +---+                | S |               |   |
                /-----UDP-----| G |------TCP------+---+
         +---+ /              +---+
         | c |/
         +---+
         example.com       example.net           example.net
         DOTS clients      DOTS gateway (DOTSG)  DOTS server
```

Figure 9: Server-Side Gateway without Aggregation

This document anticipates scenarios involving multiple DOTS gateways. An example is a DOTS gateway at the network client's side, and another one at the server side. The first gateway can be located at a Customer Premises Equipment (CPE) to aggregate requests from multiple DOTS clients enabled in an enterprise network. The second DOTS gateway is deployed on the provider side. This scenario can be seen as a combination of the client-side and server-side scenarios.

3. Concepts

3.1. DOTS Sessions

In order for DOTS to be effective as a vehicle for DDoS mitigation requests, one or more DOTS clients must establish ongoing communication with one or more DOTS servers. While the preconditions for enabling DOTS in or among network domains may also involve business relationships, service level agreements, or other formal or informal understandings between network operators, such considerations are out of scope for this document.

A DOTS session is established to support bilateral exchange of data between an associated DOTS client and a DOTS server. In the DOTS architecture, data is exchanged between DOTS agents over signal and data channels. As such, a DOTS session can be a DOTS signal channel session, a DOTS data channel session, or both. The DOTS server couples the DOTS signal and data channel sessions using the DOTS client identity. The DOTS session is further elaborated in the DOTS signal channel protocol defined in [I-D.ietf-dots-signal-channel] and the DOTS data channel protocol defined in [I-D.ietf-dots-data-channel].

A DOTS agent can maintain one or more DOTS sessions.

A DOTS signal channel session is associated with a single transport connection (TCP or UDP session) and an security association (a TLS or DTLS session). Similarly, a DOTS data channel session is associated with a single TCP connection and an TLS security association.

Mitigation requests created using DOTS signal channel are not bound to the DOTS signal channel session. Instead, mitigation requests are associated with a DOTS client and can be managed using different DOTS signal channel sessions.

3.1.1. Preconditions

Prior to establishing a DOTS session between agents, the owners of the networks, domains, services or applications involved are assumed to have agreed upon the terms of the relationship involved. Such agreements are out of scope for this document, but must be in place for a functional DOTS architecture.

It is assumed that as part of any DOTS service agreement, the DOTS client is provided with all data and metadata required to establish communication with the DOTS server. Such data and metadata would include any cryptographic information necessary to meet the message confidentiality, integrity and authenticity requirement (SEC-002) in [RFC8612], and might also include the pool of DOTS server addresses and ports the DOTS client should use for signal and data channel messaging.

3.1.2. Establishing the DOTS Session

With the required business agreements in place, the DOTS client initiates a DOTS session by contacting its DOTS server(s) over the signal channel and (possibly) the data channel. To allow for DOTS service flexibility, neither the order of contact nor the time interval between channel creations is specified. A DOTS client MAY establish signal channel first, and then data channel, or vice versa.

The methods by which a DOTS client receives the address and associated service details of the DOTS server are not prescribed by this document. For example, a DOTS client may be directly configured to use a specific DOTS server IP address and port, and directly provided with any data necessary to satisfy the Peer Mutual Authentication requirement (SEC-001) in [RFC8612], such as symmetric or asymmetric keys, usernames and passwords, etc. All configuration and authentication information in this scenario is provided out-of-band by the domain operating the DOTS server.

At the other extreme, the architecture in this document allows for a form of DOTS client auto-provisioning. For example, the domain operating the DOTS server or servers might provide the client domain only with symmetric or asymmetric keys to authenticate the provisioned DOTS clients. Only the keys would then be directly configured on DOTS clients, but the remaining configuration required to provision the DOTS clients could be learned through mechanisms similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

The DOTS client SHOULD successfully authenticate and exchange messages with the DOTS server over both signal and (if used) data channel as soon as possible to confirm that both channels are operational.

As described in [RFC8612] (DM-008), the DOTS client can configure preferred values for acceptable signal loss, mitigation lifetime, and heartbeat intervals when establishing the DOTS signal channel session. A DOTS signal channel session is not active until DOTS agents have agreed on the values for these DOTS session parameters, a process defined by the protocol.

Once the DOTS client begins receiving DOTS server signals, the DOTS session is active. At any time during the DOTS session, the DOTS client may use the data channel to manage aliases, manage drop- and accept-listed prefixes or addresses, leverage vendor-specific extensions, and so on. Note that unlike the signal channel, there is no requirement that the data channel remains operational in attack conditions (See Data Channel Requirements, Section 2.3 of [RFC8612]).

3.1.3. Maintaining the DOTS Session

DOTS clients and servers periodically send heartbeats to each other over the signal channel, discussed in [RFC8612] (SIG-004). DOTS agent operators SHOULD configure the heartbeat interval such that the frequency does not lead to accidental denials of service due to the overwhelming number of heartbeats a DOTS agent must field.

Either DOTS agent may consider a DOTS signal channel session terminated in the extended absence of a heartbeat from its peer agent. The period of that absence will be established in the protocol definition.

3.2. Modes of Signaling

This section examines the modes of signaling between agents in a DOTS architecture.

3.2.1. Direct Signaling

A DOTS session may take the form of direct signaling between the DOTS clients and servers, as shown in Figure 10.

```
+-------------+                            +-------------+
| DOTS client |<------signal session------>| DOTS server |
+-------------+                            +-------------+
```

Figure 10: Direct Signaling

In a direct DOTS session, the DOTS client and server are communicating directly. Direct signaling may exist inter- or intra-domain. The DOTS session is abstracted from the underlying networks or network elements the signals traverse: in direct signaling, the DOTS client and server are logically adjacent.

3.2.2. Redirected Signaling

In certain circumstances, a DOTS server may want to redirect a DOTS client to an alternative DOTS server for a DOTS signal channel session. Such circumstances include but are not limited to:

- Maximum number of DOTS signal channel sessions with clients has been reached;
- Mitigation capacity exhaustion in the mitigator with which the specific DOTS server is communicating;
- Mitigator outage or other downtime, such as scheduled maintenance;
- Scheduled DOTS server maintenance;
- Scheduled modifications to the network path between DOTS server and DOTS client.

A basic redirected DOTS signal channel session resembles the following, as shown in Figure 11.

```
      +-------------+                             +---------------+
      |             |<-(1)--- DOTS signal ------>|               |
      |             |      channel session 1      |               |
      |             |<=(2)== redirect to B ======|               |
      | DOTS client |                             | DOTS server A |
      |             |X-(4)--- DOTS signal ------X|               |
      |             |      channel session 1      |               |
      |             |                             |               |
      +-------------+                             +---------------+
             ^
             |
            (3) DOTS signal channel
             |      session 2
             v
      +---------------+
      | DOTS server B |
      +---------------+
```

Figure 11: Redirected Signaling

1. Previously established DOTS signal channel session 1 exists between a DOTS client and DOTS server A.

2. DOTS server A sends a server signal redirecting the client to DOTS server B.

3. If the DOTS client does not already have a separate DOTS signal channel session with the redirection target, the DOTS client initiates and establishes DOTS signal channel session 2 with DOTS server B.

4. Having redirected the DOTS client, DOTS server A ceases sending server signals. The DOTS client likewise stops sending client signals to DOTS server A. DOTS signal channel session 1 is terminated.

3.2.3. Recursive Signaling

DOTS is centered around improving the speed and efficiency of coordinated response to DDoS attacks. One scenario not yet discussed involves coordination among federated domains operating DOTS servers and mitigators.

In the course of normal DOTS operations, a DOTS client communicates the need for mitigation to a DOTS server, and that server initiates mitigation on a mitigator with which the server has an established service relationship. The operator of the mitigator may in turn monitor mitigation performance and capacity, as the attack being mitigated may grow in severity beyond the mitigating domain’s capabilities.

The operator of the mitigator has limited options in the event a DOTS client-requested mitigation is being overwhelmed by the severity of the attack. Out-of-scope business or service level agreements may permit the mitigating domain to drop the mitigation and let attack traffic flow unchecked to the target, but this only encourages attack escalation. In the case where the mitigating domain is the upstream service provider for the attack target, this may mean the mitigating domain and its other services and users continue to suffer the incidental effects of the attack.

A recursive signaling model as shown in Figure 12 offers an alternative. In a variation of the use case "Upstream DDoS Mitigation by an Upstream Internet Transit Provider" described in [I-D.ietf-dots-use-cases], a domain operating a DOTS server and mitigator also operates a DOTS client. This DOTS client has an established DOTS session with a DOTS server belonging to a separate administrative domain.

With these preconditions in place, the operator of the mitigator being overwhelmed or otherwise performing inadequately may request mitigation for the attack target from this separate DOTS-aware domain. Such a request recurses the originating mitigation request to the secondary DOTS server, in the hope of building a cumulative mitigation against the attack.

```
                        example.net domain
                        . . . . . . . . . . . . . . . . .
                        .    Gn                          .
          +----+    1   .  +----+       +-----------+    .
          | Cc |<--------->| Sn |~~~~~~~| Mitigator |    .
          +----+        .  +====+       |     Mn    |    .
                        .  | Cn |       +-----------+    .
        example.com     .  +----+                        .
           client       .    ^                           .
                        . . .|. . . . . . . . . . . . . .
                             |
                           2 |
                             |
                        . . .|. . . . . . . . . . . . . .
                        .    v                           .
                        .  +----+       +-----------+    .
                        .  | So |~~~~~~~| Mitigator |    .
                        .  +----+       |     Mo    |    .
                        .               +-----------+    .
                        .                                .
                        . . . . . . . . . . . . . . . . .
                        example.org domain
```

Figure 12: Recursive Signaling

In Figure 12, client Cc signals a request for mitigation across inter-domain DOTS session 1 to the DOTS server Sn belonging to the example.net domain. DOTS server Sn enables mitigation on mitigator Mn. DOTS server Sn is half of DOTS gateway Gn, being deployed logically back-to-back with DOTS client Cn, which has pre-existing inter-domain DOTS session 2 with the DOTS server So belonging to the example.org domain. At any point, DOTS server Sn MAY recurse an on-going mitigation request through DOTS client Cn to DOTS server So, in the expectation that mitigator Mo will be activated to aid in the defense of the attack target.

Recursive signaling is opaque to the DOTS client. To maximize mitigation visibility to the DOTS client, however, the recursing domain SHOULD provide recursed mitigation feedback in signals reporting on mitigation status to the DOTS client. For example, the recursing domain's DOTS server should incorporate into mitigation status messages available metrics such as dropped packet or byte counts from the recursed domain's DOTS server.

DOTS clients involved in recursive signaling must be able to withdraw requests for mitigation without warning or justification, per SIG-006 in [RFC8612].

Operators recursing mitigation requests MAY maintain the recursed mitigation for a brief, protocol-defined period in the event the DOTS client originating the mitigation withdraws its request for help, as per the discussion of managing mitigation toggling in SIG-006 of [RFC8612].

Deployment of recursive signaling may result in traffic redirection, examination and mitigation extending beyond the initial bilateral relationship between DOTS client and DOTS server. As such, client control over the network path of mitigated traffic may be reduced. DOTS client operators should be aware of any privacy concerns, and work with DOTS server operators employing recursive signaling to ensure shared sensitive material is suitably protected. Typically there is a contractual Service Level Agreement (SLA) negotiated among the DOTS client domain, the recursed domain and the recursing domain to meet the privacy requirements of the DOTS client domain and authorization for the recursing domain to request mitigation for the resources controlled by the DOTS client domain.

3.2.4. Anycast Signaling

The DOTS architecture does not assume the availability of anycast within a DOTS deployment, but neither does the architecture exclude it. Domains operating DOTS servers MAY deploy DOTS servers with an anycast Service Address as described in BCP 126 [RFC4786]. In such a deployment, DOTS clients connecting to the DOTS Service Address may be communicating with distinct DOTS servers, depending on the network configuration at the time the DOTS clients connect. Among other benefits, anycast signaling potentially offers the following:

- Simplified DOTS client configuration, including service discovery through the methods described in [RFC7094]. In this scenario, the "instance discovery" message would be a DOTS client initiating a DOTS session to the DOTS server anycast Service Address, to which the DOTS server would reply with a redirection to the DOTS server unicast address the client should use for DOTS.

- Region- or customer-specific deployments, in which the DOTS Service Addresses route to distinct DOTS servers depending on the client region or the customer network in which a DOTS client resides.

- Operational resiliency, spreading DOTS signaling traffic across the DOTS server domain’s networks, and thereby also reducing the potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

As long as network configuration remains stable, anycast DOTS signaling is to the individual DOTS client indistinct from direct signaling. However, the operational challenges inherent in anycast signaling are anything but negligible, and DOTS server operators must carefully weigh the risks against the benefits before deploying.

While the DOTS signal channel primarily operates over UDP per SIG-001 in [RFC8612], the signal channel also requires mutual authentication between DOTS agents, with associated security state on both ends.

Network instability is of particular concern with anycast signaling, as DOTS signal channels are expected to be long-lived, and potentially operating under congested network conditions caused by a volumetric DDoS attack.

For example, a network configuration altering the route to the DOTS server during active anycast signaling may cause the DOTS client to send messages to a DOTS server other than the one with which it initially established a signaling session. That second DOTS server might not have the security state of the existing session, forcing the DOTS client to initialize a new DOTS session. This challenge might in part be mitigated by use of resumption via a PSK in TLS 1.3 [RFC8446] and DTLS 1.3 [I-D.ietf-tls-dtls13] (session resumption in TLS 1.2 [RFC5246] and DTLS 1.2 [RFC6347]), but keying material must be available to all DOTS servers sharing the anycast Service Address in that case which has operational challenges of its own.

While the DOTS client will try to establish a new DOTS session with the DOTS server now acting as the anycast DOTS Service Address, the link between DOTS client and server may be congested with attack traffic, making signal session establishment difficult. In such a scenario, anycast Service Address instability becomes a sort of signal session flapping, with obvious negative consequences for the DOTS deployment.

Anycast signaling deployments similarly must also take into account active mitigations. Active mitigations initiated through a DOTS session may involve diverting traffic to a scrubbing center. If the DOTS session flaps due to anycast changes as described above, mitigation may also flap as the DOTS servers sharing the anycast DOTS service address toggles mitigation on detecting DOTS session loss, depending on whether the client has configured mitigation on loss of signal (Section 3.3.3).

3.2.5. Signaling Considerations for Network Address Translation

Network address translators (NATs) are expected to be a common feature of DOTS deployments. The Middlebox Traversal Guidelines in [RFC8085] include general NAT considerations that are applicable to DOTS deployments when the signal channel is established over UDP.

Additional DOTS-specific considerations arise when NATs are part of the DOTS architecture. For example, DDoS attack detection behind a NAT will detect attacks against internal addresses. A DOTS client subsequently asked to request mitigation for the attacked scope of addresses cannot reasonably perform the task, due to the lack of externally routable addresses in the mitigation scope.

The following considerations do not cover all possible scenarios, but are meant rather to highlight anticipated common issues when signaling through NATs.

3.2.5.1. Direct Provisioning of Internal-to-External Address Mappings

Operators may circumvent the problem of translating internal addresses or prefixes to externally routable mitigation scopes by directly provisioning the mappings of external addresses to internal protected resources on the DOTS client. When the operator requests mitigation scoped for internal addresses, directly or through automated means, the DOTS client looks up the matching external addresses or prefixes, and issues a mitigation request scoped to that externally routable information.

When directly provisioning the address mappings, operators must ensure the mappings remain up to date, or risk losing the ability to request accurate mitigation scopes. To that aim, the DOTS client can rely on mechanisms such as [RFC8512] or [RFC7658] to retrieve static explicit mappings. This document does not prescribe the method by which mappings are maintained once they are provisioned on the DOTS client.

3.2.5.2. Resolving Public Mitigation Scope with Port Control Protocol (PCP)

Port Control Protocol (PCP) [RFC6887] may be used to retrieve the external addresses/prefixes and/or port numbers if the NAT function embeds a PCP server.

A DOTS client can use the information retrieved by means of PCP to feed the DOTS protocol(s) messages that will be sent to a DOTS server. These messages will convey the external addresses/prefixes as set by the NAT.

PCP also enables discovery and configuration of the lifetime of port mappings instantiated in intermediate NAT devices. Discovery of port mapping lifetimes can reduce the dependency on heartbeat messages to maintain mappings, and therefore reduce the load on DOTS servers and the network.

3.2.5.3. Resolving Public Mitigation Scope with Session Traversal Utilities (STUN)

An internal resource, e.g., a Web server, can discover its reflexive transport address through a STUN Binding request/response transaction, as described in [I-D.ietf-tram-stunbis]. After learning its reflexive transport address from the STUN server, the internal resource can export its reflexive transport address and internal transport address to the DOTS client, thereby enabling the DOTS client to request mitigation with the correct external scope, as depicted in Figure 13. The mechanism for providing the DOTS client with the reflexive transport address and internal transport address is unspecified in this document.

In order to prevent an attacker from modifying the STUN messages in transit, the STUN client and server must use the message-integrity mechanism discussed in Section 9 of [I-D.ietf-tram-stunbis] or use STUN over DTLS [RFC7350] or use STUN over TLS. If the STUN client is behind a NAT that performs Endpoint-Dependent Mapping [RFC5128], the internal service cannot provide the DOTS client with the reflexive transport address discovered using STUN. The behavior of a NAT between the STUN client and the STUN server could be discovered using the experimental techniques discussed in [RFC5780], but note that there is currently no standardized way for a STUN client to reliably determine if it is behind a NAT that performs Endpoint-Dependent Mapping.

```
               Binding         Binding
   +--------+  request  +---+  request  +--------+
   |  STUN  |<----------| N |<----------|  STUN  |
   | server |           | A |           | client |
   |        |---------->| T |---------->|        |
   +--------+  Binding  +---+ Binding   +--------+
               response       response    |
                                          | reflexive transport address
                                          | & internal transport address
                                          v
                                        +--------+
                                        |  DOTS  |
                                        | client |
                                        +--------+
```

Figure 13: Resolving mitigation scope with STUN

3.2.5.4. Resolving Requested Mitigation Scope with DNS

DOTS supports mitigation scoped to DNS names. As discussed in [RFC3235], using DNS names instead of IP addresses potentially avoids the address translation problem, as long as the same domain name is internally and externally resolvable. For example, a detected attack’s internal target address can be mapped to a DNS name through a reverse lookup. The DNS name returned by the reverse lookup can then be provided to the DOTS client as the external scope for mitigation. For the reverse DNS lookup, DNS Security Extensions (DNSSEC) [RFC4033] must be used where the authenticity of response is critical.

3.3. Triggering Requests for Mitigation

[RFC8612] places no limitation on the circumstances in which a DOTS client operator may request mitigation, nor does it demand justification for any mitigation request, thereby reserving operational control over DDoS defense for the domain requesting mitigation. This architecture likewise does not prescribe the network conditions and mechanisms triggering a mitigation request from a DOTS client.

However, considering selected possible mitigation triggers from an architectural perspective offers a model for alternative or unanticipated triggers for DOTS deployments. In all cases, what network conditions merit a mitigation request are at the discretion of the DOTS client operator.

The mitigation request itself is defined by DOTS, however the interfaces required to trigger the mitigation request in the following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

A DOTS client operator may manually prepare a request for mitigation, including scope and duration, and manually instruct the DOTS client to send the mitigation request to the DOTS server. In context, a manual request is a request directly issued by the operator without automated decision-making performed by a device interacting with the DOTS client. Modes of manual mitigation requests include an operator entering a command into a text interface, or directly interacting with a graphical interface to send the request.

An operator might do this, for example, in response to notice of an attack delivered by attack detection equipment or software, and the alerting detector lacks interfaces or is not configured to use available interfaces to translate the alert to a mitigation request automatically.

In a variation of the above scenario, the operator may have preconfigured on the DOTS client mitigation requests for various resources in the operator’s domain. When notified of an attack, the DOTS client operator manually instructs the DOTS client to send the relevant preconfigured mitigation request for the resources under attack.

A further variant involves recursive signaling, as described in Section 3.2.3. The DOTS client in this case is the second half of a DOTS gateway (back-to-back DOTS server and client). As in the previous scenario, the scope and duration of the mitigation request are pre-existing, but in this case are derived from the mitigation request received from a downstream DOTS client by the DOTS server. Assuming the preconditions required by Section 3.2.3 are in place, the DOTS gateway operator may at any time manually request mitigation from an upstream DOTS server, sending a mitigation request derived from the downstream DOTS client’s request.

The motivations for a DOTS client operator to request mitigation manually are not prescribed by this architecture, but are expected to include some of the following:

- Notice of an attack delivered via e-mail or alternative messaging

- Notice of an attack delivered via phone call

- Notice of an attack delivered through the interface(s) of networking monitoring software deployed in the operator’s domain

- Manual monitoring of network behavior through network monitoring software

3.3.2. Automated Conditional Mitigation Request

Unlike manual mitigation requests, which depend entirely on the DOTS client operator’s capacity to react with speed and accuracy to every detected or detectable attack, mitigation requests triggered by detected attack conditions reduce the operational burden on the DOTS client operator, and minimize the latency between attack detection and the start of mitigation.

Mitigation requests are triggered in this scenario by operator-specified network conditions. Attack detection is deployment-specific, and not constrained by this architecture. Similarly the specifics of a condition are left to the discretion of the operator, though common conditions meriting mitigation include the following:

- Detected attack exceeding a rate in packets per second (pps).

- Detected attack exceeding a rate in bytes per second (bps).

- Detected resource exhaustion in an attack target.

- Detected resource exhaustion in the local domain’s mitigator.

- Number of open connections to an attack target.

- Number of attack sources in a given attack.

- Number of active attacks against targets in the operator’s domain.

- Conditional detection developed through arbitrary statistical analysis or deep learning techniques.

- Any combination of the above.

When automated conditional mitigation requests are enabled, violations of any of the above conditions, or any additional operator-defined conditions, will trigger a mitigation request from the DOTS client to the DOTS server. The interfaces between the application detecting the condition violation and the DOTS client are implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

To maintain a DOTS signal channel session, the DOTS client and the DOTS server exchange regular but infrequent messages across the signal channel. In the absence of an attack, the probability of message loss in the signaling channel should be extremely low. Under attack conditions, however, some signal loss may be anticipated as attack traffic congests the link, depending on the attack type.

While [RFC8612] specifies the DOTS protocol be robust when signaling under attack conditions, there are nevertheless scenarios in which the DOTS signal is lost in spite of protocol best efforts. To handle such scenarios, a DOTS operator may request one or more mitigations which are triggered only when the DOTS server ceases receiving DOTS client heartbeats beyond the miss count or interval permitted by the protocol.

The impact of mitigating due to loss of signal in either direction must be considered carefully before enabling it. Attack traffic congesting links is not the only reason why signal could be lost, and as such mitigation requests triggered by signal channel degradation in either direction may incur unnecessary costs due to scrubbing traffic, adversely impact network performance and operational expense alike.

4. IANA Considerations

This document has no actions for IANA.

5. Security Considerations

This section describes identified security considerations for the DOTS architecture.

Security considerations and security requirements discussed in [RFC8612] need to be taken into account.

DOTS is at risk from three primary attack vectors: agent impersonation, traffic injection and signal blocking. These vectors may be exploited individually or in concert by an attacker to confuse, disable, take information from, or otherwise inhibit DOTS agents.

Any attacker with the ability to impersonate a legitimate DOTS client or server or, indeed, inject false messages into the stream may potentially trigger/withdraw traffic redirection, trigger/cancel mitigation activities or subvert drop-/accept-lists. From an architectural standpoint, operators MUST ensure conformance to the security requirements defined in Section 2.4 of [RFC8612] to secure data in transit. Similarly, as the received data may contain network topology, telemetry, threat and mitigation information which could be considered sensitive in certain environment, it SHOULD be protected at rest per required local policy.

DOTS agents MUST perform mutual authentication to ensure authenticity of each other and DOTS servers MUST verify that the requesting DOTS client is authorized to request mitigation for specific target resources (see Section 2.2.2).

An MITM attacker can intercept and drop packets, preventing the DOTS peers from receiving some or all of the DOTS messages, automated mitigation on loss of signal can be used as a countermeasure but with risks discussed in Section 3.3.3.

An attacker with control of a DOTS client may negatively influence network traffic by requesting and withdrawing requests for mitigation for particular prefixes, leading to route or DNS flapping. DOTS operators should carefully monitor and audit DOTS clients to detect misbehavior and deter misuse.

Any attack targeting the availability of DOTS servers may disrupt the ability of the system to receive and process DOTS signals resulting in failure to fulfill a mitigation request. DOTS servers MUST be given adequate protections, in accordance with best current practices for network and host security.

6. Contributors

Mohamed Boucadair
Orange

mohamed.boucadair@orange.com

Christopher Gray Christopher_Gray3@cable.comcast.com

7. Acknowledgments

Thanks to Matt Richardson, Roman Danyliw, Frank Xialiang, Roland Dobbins, Wei Pan, Kaname Nishizuka, Jon Shallow, Paul Kyzivat, Warren Kumari, Benjamin Kaduk, and Mohamed Boucadair for their comments and suggestions.

Special thanks to Roman Danyliw for the AD review.

8. References

8.1. 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>.

[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>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[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>.

[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>.

[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>.

8.2. Informative References

[I-D.ietf-acme-ip] Shoemaker, R., "ACME IP Identifier Validation Extension", draft-ietf-acme-ip-08 (work in progress), October 2019.

[I-D.ietf-dots-data-channel] Boucadair, M. and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-31 (work in progress), July 2019.

[I-D.ietf-dots-signal-channel]
Reddy.K, T., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-41 (work in progress), January 2020.

[I-D.ietf-dots-use-cases]
Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-20 (work in progress), September 2019.

[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-dtls13-34 (work in progress), November 2019.

[I-D.ietf-tram-stunbis]
Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, "Session Traversal Utilities for NAT (STUN)", draft-ietf-tram-stunbis-21 (work in progress), March 2019.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235, DOI 10.17487/RFC3235, January 2002, <https://www.rfc-editor.org/info/rfc3235>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[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>.

[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>.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <https://www.rfc-editor.org/info/rfc5128>.

[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>.

[RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, DOI 10.17487/RFC5780, May 2010, <https://www.rfc-editor.org/info/rfc5780>.

[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>.

[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>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <https://www.rfc-editor.org/info/rfc7092>.

[RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil, "Architectural Considerations of IP Anycast", RFC 7094, DOI 10.17487/RFC7094, January 2014, <https://www.rfc-editor.org/info/rfc7094>.

[RFC7350] Petit-Huguenin, M. and G. Salgueiro, "Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN)", RFC 7350, DOI 10.17487/RFC7350, August 2014, <https://www.rfc-editor.org/info/rfc7350>.

[RFC7658] Perreault, S., Tsou, T., Sivakumar, S., and T. Taylor, "Deprecation of MIB Module NAT-MIB: Managed Objects for Network Address Translators (NATs)", RFC 7658, DOI 10.17487/RFC7658, October 2015, <https://www.rfc-editor.org/info/rfc7658>.

[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>.

[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>.

[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>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/info/rfc8555>.

Authors' Addresses

Andrew Mortensen (editor)
Forcepoint
United States

EMail: andrewmortensen@gmail.com

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

EMail: kondtir@gmail.com

Flemming Andreasen
Cisco
United States

EMail: fandreas@cisco.com

Nik Teague
Iron Mountain
United States

EMail: nteague@ironmountain.co.uk

Rich Compton
Charter

EMail: Rich.Compton@charter.com

DOTS
Internet-Draft
Intended status: Informational
Expires: May 3, 2017

A. Mortensen
Arbor Networks, Inc.
R. Moskowitz
HTT Consulting
T. Reddy
Cisco Systems, Inc.

October 30, 2016

Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
draft-ietf-dots-requirements-03

Abstract

This document defines the requirements for the Distributed Denial of Service (DDoS) Open Threat Signaling (DOTS) protocols coordinating attack response against DDoS attacks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Context and Motivation

Distributed Denial of Service (DDoS) attacks continue to plague networks around the globe, from Tier-1 service providers on down to enterprises and small businesses. Attack scale and frequency similarly have continued to increase, in part as a result of software vulnerabilities leading to reflection and amplification attacks. Once staggering attack traffic volume is now the norm, and the impact of larger-scale attacks attract the attention of international press agencies.

The greater impact of contemporary DDoS attacks has led to increased focus on coordinated attack response. Many institutions and enterprises lack the resources or expertise to operate on-premise attack mitigation solutions themselves, or simply find themselves constrained by local bandwidth limitations. To address such gaps, security service providers have begun to offer on-demand traffic scrubbing services, which aim to separate the DDoS traffic from legitimate traffic and forward only the latter. Today each such service offers its own interface for subscribers to request attack mitigation, tying subscribers to proprietary implementations while also limiting the subset of network elements capable of participating in the attack response. As a result of incompatibility across services, attack responses may be fragmentary or otherwise incomplete, leaving key players in the attack path unable to assist in the defense.

The lack of a common method to coordinate a real-time response among involved actors and network domains inhibits the speed and effectiveness of DDoS attack mitigation. This document describes the required characteristics of a DOTS protocol enabling requests for DDoS attack mitigation, reducing attack impact and leading to more efficient defensive strategies.

DOTS communicates the need for defensive action in anticipation of or in response to an attack, but does not dictate the form any defensive action takes. DOTS supplements calls for help with pertinent details about the detected attack, allowing entities participating in DOTS to form ad hoc, adaptive alliances against DDoS attacks as described in the DOTS use cases [I-D.ietf-dots-use-cases]. The requirements in this document are derived from those use cases and [I-D.ietf-dots-architecture].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic originating from multiple sources are directed at a target on a network. DDoS attacks are intended to cause a negative impact on the availability of servers, services, applications, and/or other functionality of an attack target. Denial-of-service considerations are discussed in detail in [RFC4732].

DDoS attack target: A network connected entity with a finite set of resources, such as network bandwidth, memory or CPU, that is the focus of a DDoS attack. Potential targets include network elements, servers, and services.

DDoS attack telemetry: Collected behavioral characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions taken to recognize and filter out DDoS attack traffic while passing legitimate traffic to the attack target.

Mitigation: A set of countermeasures enforced against traffic destined for the target or targets of a detected or reported DDoS attack, where countermeasure enforcement is managed by an entity in the network path between attack sources and the attack target. Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of performing mitigation of a detected or reported DDoS attack. For the purposes of this document, this entity is a black box capable of mitigation, making no assumptions about availability or design of countermeasures, nor about the programmable interface between this entity and other network elements. The mitigator and DOTS server are assumed to belong to the same administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to messages from DOTS clients. The DOTS server SHOULD enable mitigation on behalf of the DOTS client, if requested, by communicating the DOTS client’s request to the mitigator and returning selected mitigator feedback to the requesting DOTS client. A DOTS server MAY also be a mitigator.

DOTS agent: Any DOTS-aware software module capable of participating in a DOTS signaling session.

DOTS gateway: A logical DOTS agent resulting from the logical concatenation of a DOTS server and a DOTS client, analogous to a SIP Back-to-Back User Agent (B2BUA) [RFC3261]. DOTS gateways are discussed in detail in [I-D.ietf-dots-architecture].

Signal channel: A bidirectional, mutually authenticated communication channel between DOTS agents characterized by resilience even in conditions leading to severe packet loss, such as a volumetric DDoS attack causing network congestion.

DOTS signal: A concise authenticated status/control message transmitted between DOTS agents, used to indicate client’s need for mitigation, as well as to convey the status of any requested mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal channel, used as a keep-alive and to measure peer health.

Client signal: A message sent from a DOTS client to a DOTS server over the signal channel, indicating the DOTS client's need for mitigation, as well as the scope of any requested mitigation, optionally including additional attack details to supplement server-initiated mitigation.

Server signal: A message sent from a DOTS server to a DOTS client over the signal channel. Note that a server signal is not a response to client signal, but a DOTS server-initiated status message sent to DOTS clients with which the server has established signaling sessions.

Data channel: A secure communication layer between DOTS clients and DOTS servers used for infrequent bulk exchange of data not easily or appropriately communicated through the signal channel under attack conditions.

Filter: A policy matching a network traffic flow or set of flows and rate-limiting or discarding matching traffic.

Blacklist: A filter list of addresses, prefixes and/or other identifiers indicating sources from which traffic should be blocked, regardless of traffic content.

Whitelist: A list of addresses, prefixes and/or other identifiers from indicating sources from which traffic should always be allowed, regardless of contradictory data gleaned in a detected attack.

Multi-homed DOTS client: A DOTS client exchanging messages with multiple DOTS servers, each in a separate administrative domain.

2. Requirements

This section describes the required features and characteristics of the DOTS protocol.

DOTS is an advisory protocol. An active DDoS attack against the entity controlling the DOTS client need not be present before establishing DOTS communication between DOTS agents. Indeed, establishing a relationship with peer DOTS agents during normal network conditions provides the foundation for more rapid attack response against future attacks, as all interactions setting up DOTS, including any business or service level agreements, are already complete.

DOTS must at a minimum make it possible for a DOTS client to request a DOTS server’s aid in mounting a coordinated defense against a suspected attack, signaling within or between domains as requested by local operators. DOTS clients should similarly be able to withdraw aid requests. DOTS requires no justification from DOTS clients for requests for help, nor do DOTS clients need to justify withdrawing help requests: the decision is local to the DOTS clients’ domain. Regular feedback between DOTS clients and DOTS server supplement the defensive alliance by maintaining a common understanding of DOTS peer health and activity. Bidirectional communication between DOTS clients and DOTS servers is therefore critical.

Yet DOTS must also work with a set of competing operational goals. On the one hand, the protocol must be resilient under extremely hostile network conditions, providing continued contact between DOTS agents even as attack traffic saturates the link. Such resiliency may be developed several ways, but characteristics such as small message size, asynchronous, redundant message delivery and minimal connection overhead (when possible given local network policy) will tend to contribute to the robustness demanded by a viable DOTS protocol. Operators of peer DOTS-enabled domains may enable quality- or class-of-service traffic tagging to increase the probability of successful DOTS signal delivery, but DOTS requires no such policies be in place. The DOTS solution indeed must be viable especially in their absence.

On the other hand, DOTS must include protections ensuring message confidentiality, integrity and authenticity to keep the protocol from becoming another vector for the very attacks it’s meant to help fight off. DOTS clients must be able to authenticate DOTS servers, and vice versa, for DOTS to operate safely, meaning the DOTS agents must have a way to negotiate and agree upon the terms of protocol security. Attacks against the transport protocol should not offer a means of attack against the message confidentiality, integrity and authenticity.

The DOTS server and client must also have some common method of defining the scope of any mitigation performed by the mitigator, as well as making adjustments to other commonly configurable features, such as listen ports, exchanging black- and white-lists, and so on.

Finally, DOTS should provide sufficient extensibility to meet local, vendor or future needs in coordinated attack defense, although this consideration is necessarily superseded by the other operational requirements.

2.1. General Requirements

GEN-001 Extensibility: Protocols and data models developed as part of DOTS MUST be extensible in order to keep DOTS adaptable to operational and proprietary DDoS defenses. Future extensions MUST be backward compatible.

GEN-002 Resilience and Robustness: The signaling protocol MUST be designed to maximize the probability of signal delivery even under the severely constrained network conditions imposed by particular attack traffic. The protocol MUST be resilient, that is, continue operating despite message loss and out-of-order or redundant message delivery.

GEN-003 Bidirectionality: To support peer health detection, to maintain an open signal channel, and to increase the probability of signal delivery during attack, the signal channel MUST be bidirectional, with client and server transmitting signals to each other at regular intervals, regardless of any client request for mitigation. Unidirectional messages MUST be supported within the bidirectional signal channel to allow for unsolicited message delivery, enabling asynchronous notifications between agents.

GEN-004 Sub-MTU Message Size: To avoid message fragmentation and the consequently decreased probability of message delivery, signaling protocol message size MUST be kept under signaling path Maximum Transmission Unit (MTU), including the byte overhead of any encapsulation, transport headers, and transport- or message-level security.

GEN-005 Bulk Data Exchange: Infrequent bulk data exchange between DOTS agents can also significantly augment attack response coordination, permitting such tasks as population of black- or white-listed source addresses; address or prefix group aliasing; exchange of incident reports; and other hinting or configuration supplementing attack response.

As the resilience requirements for the DOTS signal channel mandate small signal message size, a separate, secure data channel utilizing a reliable transport protocol MUST be used for bulk data exchange.

2.2. Operational Requirements

OP-001 Use of Common Transport Protocols: DOTS MUST operate over common widely deployed and standardized transport protocols. While the User Datagram Protocol (UDP) [RFC0768] SHOULD be used for the signal channel, the Transmission Control Protocol (TCP)

[RFC0793] MAY be used if necessary due to network policy or middlebox capabilities or configurations. The data channel MUST use a reliable transport; see Section 2.3 below.

OP-002 Session Health Monitoring: Peer DOTS agents MUST regularly send heartbeats to each other after mutual authentication in order to keep the DOTS session active. A session MUST be considered active until a DOTS agent explicitly ends the session, or either DOTS agent fails to receive heartbeats from the other after a mutually agreed upon timeout period has elapsed.

OP-003 Session Redirection: In order to increase DOTS operational flexibility and scalability, DOTS servers SHOULD be able to redirect DOTS clients to another DOTS server at any time. Due to the decreased probability of DOTS server signal delivery due to link congestion, it is RECOMMENDED DOTS servers avoid redirecting while mitigation is enabled during an active attack against a target in the DOTS client’s domain. Either the DOTS servers have to fate-share the security state, the client MUST have separate security state with each potential redirectable server, or be able to negotiate new state as part of redirection.

OP-004 Mitigation Status: DOTS MUST provide a means to report the status of an action requested by a DOTS client. In particular, DOTS clients MUST be able to request or withdraw a request for mitigation from the DOTS server. The DOTS server MUST acknowledge a DOTS client’s request to withdraw from coordinated attack response in subsequent signals, and MUST cease mitigation activity as quickly as possible. However, a DOTS client rapidly toggling active mitigation may result in undesirable side-effects for the network path, such as route or DNS [RFC1034] flapping. A DOTS server therefore MAY continue mitigating for a mutually negotiated period after receiving the DOTS client’s request to stop.

A server MAY refuse to engage in coordinated attack response with a client. To make the status of a client’s request clear, the server MUST indicate in server signals whether client-initiated mitigation is active. When a client-initiated mitigation is active, and threat handling details such as mitigation scope and statistics are available to the server, the server SHOULD include those details in server signals sent to the client. DOTS clients SHOULD take mitigation statistics into account when deciding whether to request the DOTS server cease mitigation.

OP-005 Mitigation Lifetime: A DOTS client SHOULD indicate the desired lifetime of any mitigation requested from the DOTS server. As DDoS attack duration is unpredictable, the DOTS client SHOULD be able to extend mitigation lifetime with periodic renewed requests for help. When the mitigation lifetime comes to an end, the DOTS server SHOULD delay session termination for a protocol-defined grace period to allow for delivery of delayed mitigation renewals over the signal channel. After the grace period elapses, the DOTS server MAY terminate the session at any time.

If a DOTS client does not include a mitigation lifetime in requests for help sent to the DOTS server, the DOTS server will use a reasonable default as defined by the protocol. As above, the DOTS client MAY extend a current mitigation request’s lifetime trivially with renewed requests for help.

A DOTS client MAY also request an indefinite mitigation lifetime, enabling architectures in which the mitigator is always in the traffic path to the resources for which the DOTS client is requesting protection. DOTS servers MAY refuse such requests for any reason. The reasons themselves are not in scope.

OP-006 Mitigation Scope: DOTS clients MUST indicate the desired scope of any mitigation, for example by using Classless Internet Domain Routing (CIDR) [RFC1518],[RFC1519] prefixes, [RFC2373] for IPv6 [RFC2460] prefixes, the length/prefix convention established in the Border Gateway Protocol (BGP) [RFC4271], SIP URIs [RFC3261], E.164 numbers, DNS names, or by a resource group alias agreed upon with the server through the data channel.

If there is additional information available narrowing the scope of any requested attack response, such as targeted port range, protocol, or service, DOTS clients SHOULD include that information in client signals. DOTS clients MAY also include additional attack details. Such supplemental information is OPTIONAL, and DOTS servers MAY ignore it when enabling countermeasures on the mitigator.

As an active attack evolves, clients MUST be able to adjust as necessary the scope of requested mitigation by refining the scope of resources requiring mitigation.

OP-007 Mitigation Efficacy: When a mitigation request by a DOTS client is active, DOTS clients SHOULD transmit a metric of perceived mitigation efficacy to the DOTS server, per "Automatic or Operator-Assisted CPE or PE Mitigators Request Upstream DDoS Mitigation Services" in [I-D.ietf-dots-use-cases]. DOTS servers MAY use the efficacy metric to adjust countermeasures activated on a mitigator on behalf of a DOTS client.

OP-008 Conflict Detection and Notification: Multiple DOTS clients controlled by a single administrative entity may send conflicting mitigation requests for pool of protected resources , as a result of misconfiguration, operator error, or compromised DOTS clients. DOTS servers attempting to honor conflicting requests may flap network route or DNS information, degrading the networks attempting to participate in attack response with the DOTS clients. DOTS servers SHALL detect such conflicting requests, and SHALL notify the DOTS clients in conflict. The notification SHOULD indicate the nature and scope of the conflict, for example, the overlapping prefix range in a conflicting mitigation request.

OP-009: Network Address Translator Traversal: The DOTS protocol MUST operate over networks in which Network Address Translation (NAT) is deployed. As UDP is the recommended transport for the DOTS signal channel, all considerations in "Middlebox Traversal Guidelines" in [RFC5405] apply to DOTS. Regardless of transport, DOTS protocols MUST follow established best common practices (BCPs) for NAT traversal.

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges between DOTS agents. Unlike the signal channel, which must operate nominally even when confronted with signal degradation due to packet loss, the data channel is not expected to be constructed to deal with attack conditions. As the primary function of the data channel is data exchange, a reliable transport is required in order for DOTS agents to detect data delivery success or failure.

The data channel must be extensible. We anticipate the data channel will be used for such purposes as configuration or resource discovery. For example, a DOTS client may submit to the DOTS server a collection of prefixes it wants to refer to by alias when requesting mitigation, to which the server would respond with a success status and the new prefix group alias, or an error status and message in the event the DOTS client’s data channel request failed. The transactional nature of such data exchanges suggests a separate set of requirements for the data channel, while the potentially sensitive content sent between DOTS agents requires extra precautions to ensure data privacy and authenticity.

DATA-001 Reliable transport: Messages sent over the data channel MUST be delivered reliably, in order sent.

DATA-002 Data privacy and integrity: Transmissions over the data channel are likely to contain operationally or privacy-sensitive information or instructions from the remote DOTS agent. Theft or modification of data channel transmissions could lead to information leaks or malicious transactions on behalf of the sending agent (see Section 4 below). Consequently data sent over the data channel MUST be encrypted and authenticated using current industry best practices. DOTS servers MUST enable means to prevent leaking operationally or privacy-sensitive data. Although administrative entities participating in DOTS may detail what data may be revealed to third-party DOTS agents, such considerations are not in scope for this document.

DATA-003 Resource Configuration: To help meet the general and operational requirements in this document, DOTS server implementations MUST provide an interface to configure resource identifiers, as described in OP-007. DOTS server implementations MAY expose additional configurability. Additional configurability is implementation-specific.

DATA-004 Black- and whitelist management: DOTS servers SHOULD provide methods for DOTS clients to manage black- and white-lists of traffic destined for resources belonging to a client.

For example, a DOTS client should be able to create a black- or whitelist entry; retrieve a list of current entries from either list; update the content of either list; and delete entries as necessary.

How the DOTS server determines client ownership of address space is not in scope.

2.4. Security requirements

DOTS must operate within a particularly strict security context, as an insufficiently protected signal or data channel may be subject to abuse, enabling or supplementing the very attacks DOTS purports to mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate each other before a DOTS session is considered valid. The method of authentication is not specified, but should follow current industry best practices with respect to any cryptographic mechanisms to authenticate the remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS protocols MUST take steps to protect the confidentiality, integrity and authenticity of messages sent between client and server. While specific transport- and message-level security options are not specified, the protocols MUST follow current industry best practices for encryption and message authentication.

In order for DOTS protocols to remain secure despite advancements in cryptanalysis and traffic analysis, DOTS agents MUST be able to negotiate the terms and mechanisms of protocol security, subject to the interoperability and signal message size requirements above.

While the interfaces between downstream DOTS server and upstream DOTS client within a DOTS gateway are implementation-specific, those interfaces nevertheless MUST provide security equivalent to that of the signaling sessions bridged by gateways in the signaling path. For example, when a DOTS gateway consisting of a DOTS server and DOTS client is running on the same logical device, they must be within the same process security boundary.

SEC-003 Message Replay Protection: In order to prevent a passive attacker from capturing and replaying old messages, DOTS protocols MUST provide a method for replay detection.

2.5. Data Model Requirements

The value of DOTS is in standardizing a mechanism to permit elements, networks or domains under or under threat of DDoS attack to request aid mitigating the effects of any such attack. A well-structured DOTS data model is therefore critical to the development of a successful DOTS protocol.

DM-001: Structure: The data model structure for the DOTS protocol may be described by a single module, or be divided into related collections of hierarchical modules and sub-modules. If the data model structure is split across modules, those distinct modules MUST allow references to describe the overall data model’s structural dependencies.

DM-002: Versioning: To ensure interoperability between DOTS protocol implementations, data models MUST be versioned. The version number of the initial data model SHALL be 1. Each published change to the initial published DOTS data model SHALL increment the data model version by 1.

How the protocol represents data model versions is not defined in this document.

DM-003: Mitigation Status Representation: The data model MUST provide the ability to represent a request for mitigation and the withdrawal of such a request. The data model MUST also support a representation of currently requested mitigation status, including failures and their causes.

DM-004: Mitigation Scope Representation: The data model MUST support representation of a requested mitigation’s scope. As mitigation scope may be represented in several different ways, per OP-006 above, the data model MUST be capable of flexible representation of mitigation scope.

DM-005: Mitigation Lifetime Representation: The data model MUST support representation of a mitigation request’s lifetime, including mitigations with no specified end time.

DM-006: Mitigation Efficacy Representation: The data model MUST support representation of a DOTS client’s understanding of the efficacy of a mitigation enabled through a mitigation request.

DM-007: Acceptable Signal Loss Representation: The data model MUST be able to represent the DOTS agent’s preference for acceptable signal loss when establishing a signaling session, as described in GEN-002.

DM-008: Heartbeat Interval Representation: The data model MUST be able to represent the DOTS agent’s preferred heartbeat interval, which the client may include when establishing the signal channel, as described in OP-002.

DM-009: Relationship to Transport: The DOTS data model MUST NOT depend on the specifics of any transport to represent fields in the model.

3. Congestion Control Considerations

3.1. Signal Channel

As part of a protocol expected to operate over links affected by DDoS attack traffic, the DOTS signal channel MUST NOT contribute significantly to link congestion. To meet the operational requirements above, DOTS signal channel implementations MUST support UDP. However, UDP when deployed naively can be a source of network congestion, as discussed in [RFC5405]. Signal channel implementations using UDP MUST therefore include a congestion control mechanism. The form of that congestion control is implementation-specific.

Signal channel implementations using TCP may rely on built-in TCP congestion control support.

3.2. Data Channel

As specified in DATA-001, the data channel requires reliable, in-order message delivery. Data channel implementations using TCP may rely on the TCP implementation’s built-in congestion control mechanisms.

4. Security Considerations

DOTS is at risk from three primary attacks:

- DOTS agent impersonation
- Traffic injection
- Signaling blocking

The DOTS protocol MUST be designed for minimal data transfer to address the blocking risk. Impersonation and traffic injection mitigation can be managed through current secure communications best practices. See Section 2.4 above for a detailed discussion.

5. Contributors

Med Boucadair
Orange

mohamed.boucadair@orange.com

Flemming Andreasen:
Cisco Systems, Inc.

fandreas@cisco.com

6. Acknowledgments

Thanks to Roman Danyliw and Matt Richardson for careful reading and feedback.

7. Change Log

7.1. 03 revision

2016-10-30

- Extended SEC-003 to require secure interfaces within DOTS gateways.

- Changed DATA-003 to Resource Configuration, delegating control of acceptable signal loss, heartbeat intervals, and mitigation lifetime to DOTS client.

- Added data model requirements reflecting client control over the above.

7.2. 02 revision

7.3. 01 revision

2016-03-21

- Reconciled terminology with -00 revision of [I-D.ietf-dots-use-cases].

- Terminology clarification based on working group feedback.

- Moved security-related requirements to separate section.

- Made resilience/robustness primary general requirement to align with charter.

- Clarified support for unidirectional communication within the bidirectional signal channel.

- Added proposed operational requirement to support session redirection.

- Added proposed operational requirement to support conflict notification.

- Added proposed operational requirement to support mitigation lifetime in mitigation requests.

- Added proposed operational requirement to support mitigation efficacy reporting from DOTS clients.

- Added proposed operational requirement to cache lookups of all kinds.

- Added proposed operational requirement regarding NAT traversal.

- Removed redundant mutual authentication requirement from data channel requirements.

7.4. 00 revision

2015-10-15

7.5. Initial revision

2015-09-24 Andrew Mortensen

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

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

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

8.2. Informative References

[I-D.ietf-dots-architecture] Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-00 (work in progress), July 2016.

[I-D.ietf-dots-use-cases] Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., and L. Xia, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-01 (work in progress), March 2016.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1518] Rekhter, Y. and T. Li, "An Architecture for IP Address Allocation with CIDR", RFC 1518, DOI 10.17487/RFC1518, September 1993, <http://www.rfc-editor.org/info/rfc1518>.

[RFC1519] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, DOI 10.17487/RFC1519, September 1993, <http://www.rfc-editor.org/info/rfc1519>.

[RFC2373] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, DOI 10.17487/RFC2373, July 1998, <http://www.rfc-editor.org/info/rfc2373>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[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, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4732] Handley, M., Ed., Rescorla, E., Ed., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, DOI 10.17487/RFC4732, December 2006, <http://www.rfc-editor.org/info/rfc4732>.

Authors' Addresses

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

Email: amortensen@arbor.net

Robert Moskowitz
HTT Consulting
Oak Park, MI 42837
United States

Email: rgm@htt-consult.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

DOTS   A. Mortensen
Internet-Draft   Arbor Networks
Intended status: Informational   T. Reddy
Expires: September 24, 2019   McAfee
                                                            R. Moskowitz
                                                                  Huawei
                                                          March 23, 2019

Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
draft-ietf-dots-requirements-22

Abstract

This document defines the requirements for the Distributed Denial of Service (DDoS) Open Threat Signaling (DOTS) protocols enabling coordinated response to DDoS attacks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Context and Motivation

Distributed Denial of Service (DDoS) attacks afflict networks connected to the Internet, plaguing network operators at service providers and enterprises around the world. High-volume attacks saturating inbound links are now common, as attack scale and frequency continue to increase.

The prevalence and impact of these DDoS attacks has led to an increased focus on coordinated attack response. However, many enterprises lack the resources or expertise to operate on-premises attack mitigation solutions themselves, or are constrained by local bandwidth limitations. To address such gaps, service providers have begun to offer on-demand traffic scrubbing services, which are designed to separate the DDoS attack traffic from legitimate traffic and forward only the latter.

Today, these services offer proprietary interfaces for subscribers to request attack mitigation. Such proprietary interfaces tie a subscriber to a service and limit the abilities of network elements that would otherwise be capable of participating in attack mitigation. As a result of signaling interface incompatibility, attack responses may be fragmented or otherwise incomplete, leaving operators in the attack path unable to assist in the defense.

A standardized method to coordinate a real-time response among involved operators will increase the speed and effectiveness of DDoS attack mitigation, and reduce the impact of these attacks. This document describes the required characteristics of protocols that enable attack response coordination and mitigation of DDoS attacks.

DDoS Open Threat Signaling (DOTS) communicates the need for defensive action in anticipation of or in response to an attack, but does not dictate the implementation of these actions. The DOTS use cases are discussed in [I-D.ietf-dots-use-cases] and the DOTS architecture is discussed in [I-D.ietf-dots-architecture].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174], when, and only when, they appear in all capitals. These capitalized words are used to signify the requirements for DOTS protocols design.

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic originating from multiple sources is directed at a target on a network. DDoS attacks are intended to cause a negative impact on the availability and/or other functionality of an attack target. Denial-of-service considerations are discussed in detail in [RFC4732].

DDoS attack target: A network connected entity that is the target of a DDoS attack. Potential targets include (but are not limited to) network elements, network links, servers, and services.

DDoS attack telemetry: Collected measurements and behavioral characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions focused on recognizing and filtering out specific types of DDoS attack traffic while passing legitimate traffic to the attack target. Distinct countermeasures can be layered to defend against attacks combining multiple DDoS attack types.

Mitigation: A set of countermeasures enforced against traffic destined for the target or targets of a detected or reported DDoS attack, where countermeasure enforcement is managed by an entity in the network path between attack sources and the attack target. Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of performing mitigation of a detected or reported DDoS attack. The means by which this entity performs these mitigations and how they are requested of it are out of scope for this document. The mitigator and DOTS server receiving a mitigation request are assumed to belong to the same administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to messages from DOTS clients. The DOTS server enables mitigation on behalf of the DOTS client, if requested, by communicating the DOTS client's request to the mitigator and returning selected mitigator feedback to the requesting DOTS client.

DOTS agent: Any DOTS-aware software module capable of participating in a DOTS signal or data channel. It can be a DOTS client, DOTS server, or, as a logical agent, a DOTS gateway.

DOTS gateway: A DOTS-aware software module resulting from the logical concatenation of the functionality of a DOTS server and a DOTS client into a single DOTS agent. This functionality is analogous to a Session Initiation Protocol (SIP) [RFC3261] Back-to-Back User Agent (B2BUA) [RFC7092]. A DOTS gateway has a client-facing side, which behaves as a DOTS server for downstream clients, and a server-facing side, which performs the role of DOTS client for upstream DOTS servers. Client-domain DOTS gateways are DOTS gateways that are in the DOTS client's domain, while server-domain DOTS gateways denote DOTS gateways that are in the DOTS server's domain. A DOTS gateway may terminate multiple discrete DOTS client connections and may aggregate these into a single or multiple connections. DOTS gateways are described further in [I-D.ietf-dots-architecture].

Signal channel: A bidirectional, mutually authenticated communication channel between DOTS agents that is resilient even in conditions leading to severe packet loss, such as a volumetric DDoS attack causing network congestion.

DOTS signal: A status/control message transmitted over the authenticated signal channel between DOTS agents, used to indicate the client’s need for mitigation, or to convey the status of any requested mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal channel, used as a keep-alive and to measure peer health.

Data channel: A bidirectional, mutually authenticated communication channel between two DOTS agents used for infrequent but reliable bulk exchange of data not easily or appropriately communicated through the signal channel. Reliable bulk data exchange may not function well or at all during attacks causing network congestion. The data channel is not expected to operate in such conditions.

Filter: A specification of a matching network traffic flow or set of flows. The filter will typically have a policy associated with it, e.g., rate-limiting or discarding matching traffic [RFC4949].

Drop-list: A list of filters indicating sources from which traffic should be blocked, regardless of traffic content.

Accept-list: A list of filters indicating sources from which traffic should always be allowed, regardless of contradictory data gleaned in a detected attack.

Multi-homed DOTS client: A DOTS client exchanging messages with multiple DOTS servers, each in a separate administrative domain.

2. Requirements

The expected layout and interactions amongst DOTS entities is described in the DOTS Architecture [I-D.ietf-dots-architecture].

The goal of the DOTS requirements specification is to specify the requirements for DOTS signal channel and data channel protocols that have different application and transport layer requirements. This section describes the required features and characteristics of the DOTS protocols.

The goal of DOTS protocols is to enable and manage mitigation on behalf of a network domain or resource which is or may become the focus of a DDoS attack. An active DDoS attack against the entity controlling the DOTS client need not be present before establishing a communication channel between DOTS agents. Indeed, establishing a relationship with peer DOTS agents during normal network conditions provides the foundation for more rapid attack response against future attacks, as all interactions setting up DOTS, including any business or service level agreements, are already complete. Reachability information of peer DOTS agents is provisioned to a DOTS client using a variety of manual or dynamic methods. Once a relationship between DOTS agents is established, regular communication between DOTS clients and servers enables a common understanding of the DOTS agents' health and activity.

The DOTS protocol must at a minimum make it possible for a DOTS client to request aid mounting a defense against a suspected attack. This defense could be coordinated by a DOTS server and include signaling within or between domains as requested by local operators. DOTS clients should similarly be able to withdraw aid requests. DOTS requires no justification from DOTS clients for requests for help, nor do DOTS clients need to justify withdrawing help requests: the decision is local to the DOTS clients' domain. Multi-homed DOTS clients must be able to select the appropriate DOTS server(s) to which a mitigation request is to be sent. The method for selecting the appropriate DOTS server in a multi-homed environment is out of scope for this document.

DOTS protocol implementations face competing operational goals when maintaining this bidirectional communication stream. On the one hand, DOTS must include measures to ensure message confidentiality, integrity, authenticity, and replay protection to keep the protocols from becoming additional vectors for the very attacks it is meant to help fight off. On the other hand, the protocol must be resilient under extremely hostile network conditions, providing continued contact between DOTS agents even as attack traffic saturates the link. Such resiliency may be developed several ways, but characteristics such as small message size, asynchronous, redundant message delivery and minimal connection overhead (when possible given local network policy) will tend to contribute to the robustness demanded by a viable DOTS protocol. Operators of peer DOTS-enabled domains may enable quality- or class-of-service traffic tagging to increase the probability of successful DOTS signal delivery, but DOTS does not require such policies be in place, and should be viable in their absence.

The DOTS server and client must also have some standardized method of defining the scope of any mitigation, as well as managing other mitigation-related configuration.

Finally, DOTS should be sufficiently extensible to meet future needs in coordinated attack defense, although this consideration is necessarily superseded by the other operational requirements.

2.1. General Requirements

GEN-001 Extensibility: Protocols and data models developed as part of DOTS MUST be extensible in order to keep DOTS adaptable to proprietary DDoS defenses. Future extensions MUST be backward compatible. Implementations of older protocol versions MUST ignore optional information added to DOTS messages as part of newer protocol versions. Implementations of older protocol versions MUST reject DOTS messages carrying mandatory information as part of newer protocol versions.

GEN-002 Resilience and Robustness: The signaling protocol MUST be designed to maximize the probability of signal delivery even under the severely constrained network conditions caused by attack traffic. Additional means to enhance the resilience of DOTS protocols, including when multiple DOTS servers are provisioned to the DOTS clients, SHOULD be considered. The protocol MUST be resilient, that is, continue operating despite message loss and out-of-order or redundant message delivery. In support of signaling protocol robustness, DOTS signals SHOULD be conveyed over transport and application protocols not susceptible to Head of Line Blocking. These requirements are at SHOULD strength to handle middle-boxes and firewall traversal.

GEN-003 Bulk Data Exchange: Infrequent bulk data exchange between DOTS agents can also significantly augment attack response coordination, permitting such tasks as population of drop- or accept-listed source addresses; address or prefix group aliasing; exchange of incident reports; and other hinting or configuration supplementing attack response.

As the resilience requirements for the DOTS signal channel mandate small signal message size, a separate, secure data channel utilizing a reliable transport protocol MUST be used for bulk data exchange. However, reliable bulk data exchange may not be possible during attacks causing network congestion.

GEN-004 Mitigation Hinting: DOTS clients may have access to attack details which can be used to inform mitigation techniques. Example attack details might include locally collected fingerprints for an on-going attack, or anticipated or active attack focal points based on other threat intelligence. DOTS clients MAY send mitigation hints derived from attack details to DOTS servers, in the full understanding that the DOTS server MAY ignore mitigation hints. Mitigation hints MUST be transmitted across the signal channel, as the data channel may not be functional during an attack. DOTS server handling of mitigation hints is implementation-specific.

GEN-005 Loop Handling: In certain scenarios, typically involving misconfiguration of DNS or routing policy, it may be possible for communication between DOTS agents to loop. Signal and data channel implementations should be prepared to detect and terminate such loops to prevent service disruption.

2.2. Signal Channel Requirements

SIG-001 Use of Common Transport Protocols: DOTS MUST operate over common widely deployed and standardized transport protocols. While connectionless transport such as the User Datagram Protocol (UDP) [RFC0768] SHOULD be used for the signal channel, the Transmission Control Protocol (TCP) [RFC0793] MAY be used if necessary due to network policy or middlebox capabilities or configurations.

SIG-002 Sub-MTU Message Size: To avoid message fragmentation and the consequently decreased probability of message delivery over a congested link, signaling protocol message size MUST be kept under signaling Path Maximum Transmission Unit (PMTU), including the byte overhead of any encapsulation, transport headers, and transport- or message-level security. If the total message size exceeds the path MTU, the DOTS agent MUST split the message into separate messages; for example, the list of mitigation scope types could be split into multiple lists and each list conveyed in a new message.

DOTS agents can attempt to learn PMTU using the procedures discussed in [I-D.ietf-intarea-frag-fragile]. 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].

SIG-003 Bidirectionality: To support peer health detection, to maintain an active signal channel, and to increase the probability of signal delivery during an attack, the signal channel MUST be bidirectional, with client and server transmitting signals to each other at regular intervals, regardless of any client request for mitigation. The bidirectional signal channel MUST support unidirectional messaging to enable notifications between DOTS agents.

SIG-004 Channel Health Monitoring: DOTS agents MUST support exchange of heartbeat messages over the signal channel to monitor channel health. These keepalives serve the purpose to maintain any on-path NAT or Firewall bindings to avoid cryptographic handshake for new mitigation requests. The heartbeat interval during active mitigation could be negotiable based on NAT/Firewall characteristics. Absent information about the NAT/Firewall characteristics, DOTS agent needs to ensure its on-path NAT or Firewall bindings do not expire, by using the keep-alive frequency discussed in Section 3.5 of [RFC8085].

To support scenarios in which loss of heartbeat is used to trigger mitigation, and to keep the channel active, DOTS servers MUST solicit heartbeat exchanges after successful mutual authentication. When DOTS agents are exchanging heartbeats and no mitigation request is active, either agent MAY request changes to the heartbeat rate. For example, a DOTS server might want to reduce heartbeat frequency or cease heartbeat exchanges when an active DOTS client has not requested mitigation, in order to control load.

Following mutual authentication, a signal channel MUST be considered active until a DOTS agent explicitly ends the session. When no attack traffic is present, the signal channel MUST be considered active until either DOTS agent fails to receive heartbeats from the other peer after a mutually agreed upon retransmission procedure has been exhausted. Peer DOTS agents MUST regularly send heartbeats to each other while a mitigation request is active. Because heartbeat loss is much more likely during volumetric attack, DOTS agents SHOULD avoid signal channel termination when mitigation is active and heartbeats are not received by either DOTS agent for an extended period. The exception circumstances to terminate the signal channel session during active mitigation are discussed below:

* To handle possible DOTS server restart or crash, the DOTS clients MAY attempt to establish a new signal channel session, but MUST continue to send heartbeats on the current session so that the DOTS server knows the session is still alive. If the new session is successfully established, the DOTS client can terminate the current session.

* DOTS servers are assumed to have the ability to monitor the attack, using feedback from the mitigator and other available sources, and MAY use the absence of attack traffic and lack of

client heartbeats as an indication the signal channel is defunct.

SIG-005 Channel Redirection: In order to increase DOTS operational flexibility and scalability, DOTS servers SHOULD be able to redirect DOTS clients to another DOTS server at any time. DOTS clients MUST NOT assume the redirection target DOTS server shares security state with the redirecting DOTS server. DOTS clients are free to attempt abbreviated security negotiation methods supported by the protocol, such as DTLS session resumption, but MUST be prepared to negotiate new security state with the redirection target DOTS server. The redirection DOTS server and redirecting DOTS server MUST belong to the same administrative domain.

Due to the increased likelihood of packet loss caused by link congestion during an attack, DOTS servers SHOULD NOT redirect while mitigation is enabled during an active attack against a target in the DOTS client’s domain.

SIG-006 Mitigation Requests and Status: Authorized DOTS clients MUST be able to request scoped mitigation from DOTS servers. DOTS servers MUST send status to the DOTS clients about mitigation requests. If a DOTS server rejects an authorized request for mitigation, the DOTS server MUST include a reason for the rejection in the status message sent to the client.

DOTS servers MUST regularly send mitigation status updates to authorized DOTS clients which have requested and been granted mitigation. If unreliable transport is used for the signal channel protocol, due to the higher likelihood of packet loss during a DDoS attack, DOTS servers needs to send mitigation status multiple times at regular intervals following the the data transmission guidelines discussed in Section 3.1.3 of [RFC8085].

When DOTS client-requested mitigation is active, DOTS server status messages MUST include the following mitigation metrics:

- Total number of packets blocked by the mitigation

- Current number of packets per second blocked

- Total number of bytes blocked

- Current number of bytes per second blocked

DOTS clients MAY take these metrics into account when determining whether to ask the DOTS server to cease mitigation.

A DOTS client MAY withdraw a mitigation request at any time, regardless of whether mitigation is currently active. The DOTS server MUST immediately acknowledge a DOTS client's request to stop mitigation.

To protect against route or DNS flapping caused by a client rapidly toggling mitigation, and to dampen the effect of oscillating attacks, DOTS servers MAY allow mitigation to continue for a limited period after acknowledging a DOTS client's withdrawal of a mitigation request. During this period, DOTS server status messages SHOULD indicate that mitigation is active but terminating. DOTS clients MAY reverse the mitigation termination during this active-but-terminating period with a new mitigation request for the same scope. The DOTS server MUST treat this request as a mitigation lifetime extension (see SIG-007 below).

The initial active-but-terminating period is implementation- and deployment- specific, but SHOULD be sufficiently long to absorb latency incurred by route propagation. If a DOTS client refreshes the mitigation before the active-but-terminating period elapses, the DOTS server MAY increase the active-but-terminating period up to a maximum of 300 seconds (5 minutes). After 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.

SIG-007 Mitigation Lifetime: DOTS servers MUST support mitigations for a negotiated time interval, and MUST terminate a mitigation when the lifetime elapses. DOTS servers also MUST support renewal of mitigation lifetimes in mitigation requests from DOTS clients, allowing clients to extend mitigation as necessary for the duration of an attack.

DOTS servers MUST treat a mitigation terminated due to lifetime expiration exactly as if the DOTS client originating the mitigation had asked to end the mitigation, including the active-but-terminating period, as described above in SIG-005.

DOTS clients MUST include a mitigation lifetime in all mitigation requests.

DOTS servers SHOULD support indefinite mitigation lifetimes, enabling architectures in which the mitigator is always in the traffic path to the resources for which the DOTS client is requesting protection. DOTS clients MUST be prepared to not be granted mitigations with indefinite lifetimes. DOTS servers MAY refuse mitigations with indefinite lifetimes, for policy reasons.

The reasons themselves are out of scope. If the DOTS server does not grant a mitigation request with an indefinite mitigation lifetime, it MUST set the lifetime to a value that is configured locally. That value MUST be returned in a reply to the requesting DOTS client.

SIG-008 Mitigation Scope: DOTS clients MUST indicate desired mitigation scope. The scope type will vary depending on the resources requiring mitigation. All DOTS agent implementations MUST support the following required scope types:

* IPv4 prefixes [RFC4632]

* IPv6 prefixes [RFC4291][RFC5952]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS servers MUST be able to resolve domain names and (when supported) URIs. How name resolution is managed on the DOTS server is implementation-specific.

DOTS agents MUST support mitigation scope aliases, allowing DOTS clients and servers to refer to collections of protected resources by an opaque identifier created through the data channel, direct configuration, or other means. Domain name and URI mitigation scopes may be thought of as a form of scope alias, in which the addresses to which the domain name or URI resolve represent the full scope of the mitigation.

If there is additional information available narrowing the scope of any requested attack response, such as targeted port range, protocol, or service, DOTS clients SHOULD include that information in client mitigation requests. DOTS clients MAY also include additional attack details. DOTS servers MAY ignore such supplemental information when enabling countermeasures on the mitigator.

As an active attack evolves, DOTS clients MUST be able to adjust as necessary the scope of requested mitigation by refining the scope of resources requiring mitigation.

A DOTS client may obtain the mitigation scope through direct provisioning or through implementation-specific methods of discovery. DOTS clients MUST support at least one mechanism to obtain mitigation scope.

SIG-009 Mitigation Efficacy: When a mitigation request is active, DOTS clients MUST be able to transmit a metric of perceived mitigation efficacy to the DOTS server. DOTS servers MAY use the efficacy metric to adjust countermeasures activated on a mitigator on behalf of a DOTS client.

SIG-010 Conflict Detection and Notification: Multiple DOTS clients controlled by a single administrative entity may send conflicting mitigation requests as a result of misconfiguration, operator error, or compromised DOTS clients. DOTS servers in the same administrative domain attempting to honor conflicting requests may flap network route or DNS information, degrading the networks attempting to participate in attack response with the DOTS clients. DOTS servers in a single administrative domain SHALL detect such conflicting requests, and SHALL notify the DOTS clients in conflict. The notification MUST indicate the nature and scope of the conflict, for example, the overlapping prefix range in a conflicting mitigation request.

SIG-011 Network Address Translator Traversal: DOTS clients may be deployed behind a Network Address Translator (NAT), and need to communicate with DOTS servers through the NAT. DOTS protocols MUST therefore be capable of traversing NATs.

If UDP is used as the transport for the DOTS signal channel, all considerations in "Middlebox Traversal Guidelines" in [RFC8085] apply to DOTS. Regardless of transport, DOTS protocols MUST follow established best common practices established in BCP 127 for NAT traversal [RFC4787][RFC6888][RFC7857].

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges between DOTS agents. Unlike the signal channel, the data channel is not expected to be constructed to deal with attack conditions. As the primary function of the data channel is data exchange, a reliable transport is required in order for DOTS agents to detect data delivery success or failure.

The data channel provides a protocol for DOTS configuration, management. For example, a DOTS client may submit to a DOTS server a collection of prefixes it wants to refer to by alias when requesting mitigation, to which the server would respond with a success status and the new prefix group alias, or an error status and message in the event the DOTS client’s data channel request failed.

DATA-001 Reliable transport: Messages sent over the data channel MUST be delivered reliably, in order sent.

DATA-003 Resource Configuration: To help meet the general and signal channel requirements in Section 2.1 and Section 2.2, DOTS server implementations MUST provide an interface to configure resource identifiers, as described in SIG-008. DOTS server implementations MAY expose additional configurability. Additional configurability is implementation-specific.

DATA-004 Policy management: DOTS servers MUST provide methods for DOTS clients to manage drop- and accept-lists of traffic destined for resources belonging to a client.

For example, a DOTS client should be able to create a drop- or accept-list entry, retrieve a list of current entries from either list, update the content of either list, and delete entries as necessary.

How a DOTS server authorizes DOTS client management of drop- and accept-list entries is implementation-specific.

2.4. Security Requirements

DOTS must operate within a particularly strict security context, as an insufficiently protected signal or data channel may be subject to abuse, enabling or supplementing the very attacks DOTS purports to mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate each other before a DOTS signal or data channel is considered valid. The method of authentication is not specified in this document, but should follow current IETF best practices [RFC7525] with respect to any cryptographic mechanisms to authenticate the remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS protocols MUST take steps to protect the confidentiality, integrity and authenticity of messages sent between client and server. While specific transport- and message-level security options are not specified, the protocols MUST follow current IETF best practices [RFC7525] for encryption and message authentication. Client-domain DOTS gateways are more trusted than DOTS clients, while server-domain DOTS gateways and DOTS servers share the same level of trust. A security mechanism at the transport layer TLS/DTLS is thus adequate to provide security between peer DOTS agents.

In order for DOTS protocols to remain secure despite advancements in cryptanalysis and traffic analysis, DOTS agents MUST support secure negotiation of the terms and mechanisms of protocol security, subject to the interoperability and signal message size requirements in Section 2.2.

While the interfaces between downstream DOTS server and upstream DOTS client within a DOTS gateway are implementation-specific, those interfaces nevertheless MUST provide security equivalent to that of the signal channels bridged by gateways in the signaling path. For example, when a DOTS gateway consisting of a DOTS server and DOTS client is running on the same logical device, the two DOTS agents could be implemented within the same process security boundary.

SEC-003 Data privacy and integrity: Transmissions over the DOTS protocols are likely to contain operationally or privacy-sensitive information or instructions from the remote DOTS agent. Theft, modification, or replay of message transmissions could lead to information leaks or malicious transactions on behalf of the sending agent (see Section 4 below). Consequently data sent over the DOTS protocols MUST be encrypted using secure transports TLS/ DTLS. DOTS servers MUST enable means to prevent leaking operationally or privacy-sensitive data. Although administrative entities participating in DOTS may detail what data may be revealed to third-party DOTS agents, such considerations are not in scope for this document.

SEC-004 Message Replay Protection: To prevent a passive attacker from capturing and replaying old messages, and thereby potentially disrupting or influencing the network policy of the receiving DOTS agent’s domain, DOTS protocols MUST provide a method for replay detection and prevention.

Within the signal channel, messages MUST be uniquely identified such that replayed or duplicated messages can be detected and discarded. Unique mitigation requests MUST be processed at most once.

SEC-005 Authorization: DOTS servers MUST authorize all messages from DOTS clients which pertain to mitigation, configuration, filtering, or status.

DOTS servers MUST reject mitigation requests with scopes which the DOTS client is not authorized to manage.

Likewise, DOTS servers MUST refuse to allow creation, modification or deletion of scope aliases and drop-/accept-lists when the DOTS client is unauthorized.

The modes of authorization are implementation-specific.

2.5. Data Model Requirements

A well-structured DOTS data model is critical to the development of successful DOTS protocols.

DM-001 Structure: The data model structure for the DOTS protocol MAY be described by a single module, or be divided into related collections of hierarchical modules and sub-modules. If the data model structure is split across modules, those distinct modules MUST allow references to describe the overall data model's structural dependencies.

DM-002 Versioning: To ensure interoperability between DOTS protocol implementations, data models MUST be versioned. How the protocols represent data model versions is not defined in this document.

DM-003 Mitigation Status Representation: The data model MUST provide the ability to represent a request for mitigation and the withdrawal of such a request. The data model MUST also support a representation of currently requested mitigation status, including failures and their causes.

DM-004 Mitigation Scope Representation: The data model MUST support representation of a requested mitigation's scope. As mitigation scope may be represented in several different ways, per SIG-008 above, the data model MUST include extensible representation of mitigation scope.

DM-005 Mitigation Lifetime Representation: The data model MUST support representation of a mitigation request's lifetime, including mitigations with no specified end time.

DM-006 Mitigation Efficacy Representation: The data model MUST support representation of a DOTS client's understanding of the efficacy of a mitigation enabled through a mitigation request.

DM-007 Acceptable Signal Loss Representation: The data model MUST be able to represent the DOTS agent's preference for acceptable signal loss when establishing a signal channel. Measurements of loss might include, but are not restricted to, number of consecutive missed heartbeat messages, retransmission count, or request timeouts.

DM-008 Heartbeat Interval Representation: The data model MUST be able to represent the DOTS agent's preferred heartbeat interval, which the client may include when establishing the signal channel, as described in SIG-003.

DM-009 Relationship to Transport: The DOTS data model MUST NOT make any assumptions about specific characteristics of any given transport into the data model, but instead, represent the fields in the model explicitly.

3. Congestion Control Considerations

3.1. Signal Channel

As part of a protocol expected to operate over links affected by DDoS attack traffic, the DOTS signal channel MUST NOT contribute significantly to link congestion. To meet the signal channel requirements above, DOTS signal channel implementations SHOULD support connectionless transports. However, some connectionless transports when deployed naively can be a source of network congestion, as discussed in [RFC8085]. Signal channel implementations using such connectionless transports, such as UDP, therefore MUST include a congestion control mechanism.

Signal channel implementations using a IETF standard congestion-controlled transport protocol (like TCP) may rely on built-in transport congestion control support.

3.2. Data Channel

As specified in DATA-001, the data channel requires reliable, in-order message delivery. Data channel implementations using a IETF standard congestion-controlled transport protocol may rely on the transport implementation's built-in congestion control mechanisms.

4. Security Considerations

This document informs future protocols under development, and so does not have security considerations of its own. However, operators should be aware of potential risks involved in deploying DOTS. DOTS agent impersonation and signal blocking are discussed here. Additional DOTS security considerations may be found in [I-D.ietf-dots-architecture] and DOTS protocol documents.

Impersonation of either a DOTS server or a DOTS client could have catastrophic impact on operations in either domain. If an attacker has the ability to impersonate a DOTS client, that attacker can affect policy on the network path to the DOTS client's domain, up to and including instantiation of drop-lists blocking all inbound traffic to networks for which the DOTS client is authorized to request mitigation.

Similarly, an impersonated DOTS server may be able to act as a sort of malicious DOTS gateway, intercepting requests from the downstream DOTS client, and modifying them before transmission to the DOTS server to inflict the desired impact on traffic to or from the DOTS client's domain. Among other things, this malicious DOTS gateway might receive and discard mitigation requests from the DOTS client, ensuring no requested mitigation is ever applied.

To detect misuse, as detailed in Section 2.4, DOTS implementations require mutual authentication of DOTS agents in order to make agent impersonation more difficult. However, impersonation may still be possible as a result of credential theft, implementation flaws, or compromise of DOTS agents.

To detect compromised DOTS agents, DOTS operators should carefully monitor and audit DOTS agents to detect misbehavior and to deter misuse, while employing current secure network communications best practices to reduce attack surface.

Blocking communication between DOTS agents has the potential to disrupt the core function of DOTS, which is to request mitigation of active or expected DDoS attacks. The DOTS signal channel is expected to operate over congested inbound links, and, as described in Section 2.2, the signal channel protocol must be designed for minimal data transfer to reduce the incidence of signal loss.

5. IANA Considerations

This document does not require any IANA action.

6. Contributors

Mohamed Boucadair
Orange

mohamed.boucadair@orange.com

Flemming Andreasen
Cisco Systems, Inc.

fandreas@cisco.com

Dave Dolson
Sandvine

ddolson@sandvine.com

7. Acknowledgments

Thanks to Roman Danyliw, Matt Richardson, Joe Touch, Scott Bradner, Robert Sparks, Brian Weis, Benjamin Kaduk, Eric Rescorla, Alvaro Retana, Suresh Krishnan, Ben Campbell, Mirja Kuehlewind and Jon Shallow for careful reading and feedback.

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[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>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

[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>.

[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>.

8.2. Informative References

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., K, R., Teague, N., Compton, R., and c. christopher_gray3@cable.comcast.com, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-12 (work in progress), February 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-intarea-frag-fragile]
Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", draft-ietf-intarea-frag-fragile-09 (work in progress), February 2019.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <https://www.rfc-editor.org/info/rfc7092>.

[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>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[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>.

Authors' Addresses

Andrew Mortensen
Arbor Networks
2727 S. State St
Ann Arbor, MI 48104
United States

Email: andrewmortensen@gmail.com

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

Email: TirumaleswarReddy_Konda@McAfee.com

Robert Moskowitz
Huawei
Oak Park, MI 42837
United States

Email: rgm@htt-consult.com

DOTS WG
Internet-Draft
Intended status: Informational
Expires: May 21, 2017

R. Dobbins, Ed.
Arbor Networks
S. Fouant
Corero Network Security
D. Migault
Ericsson
R. Moskowitz
Huawei
N. Teague
Verisign Inc
L. Xia
Huawei
K. Nishizuka
NTT Communications

November 17, 2016

Use cases for DDoS Open Threat Signaling
draft-ietf-dots-use-cases-03.txt

Abstract

The DDoS Open Threat Signaling (DOTS) effort is intended to provide a protocol that facilitates interoperability between multivendor solutions/services. This document presents use cases to evaluate the interactions expected between the DOTS components as well as the DOTS exchanges. The purpose of the use cases is to identify the interacting DOTS component, how they collaborate and what are the types of information to be exchanged.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Currently, distributed denial-of-service (DDoS) attack mitigation solutions/services are largely based upon siloed, proprietary communications paradigms which result in vendor/service lock-in. As a side-effect, this makes the configuration, provisioning, operation, and activation of these solutions a highly manual and often time-consuming process. Additionally, coordination of multiple DDoS mitigation solutions/services simultaneously engaged in defending the same organization against DDoS attacks is fraught with both technical and process-related hurdles. This greatly increase operational complexity and often results in suboptimal DDoS attack mitigation efficacy.

The DDoS Open Threat Signaling (DOTS) effort is intended to provide a protocol that facilitates interoperability between multivendor DDoS mitigation solutions/services. As DDoS solutions/services are broadly heterogeneous among different vendors, the primary goal for DOTS is to provide a high level interaction with these DDoS solutions/services such as initiating or terminating DDoS mitigation assistance.

It should be noted that DOTS is not in and of itself intended to perform orchestration functions duplicative of the functionality being developed by the [I2NSF] WG; rather, DOTS is intended to allow devices, services, and applications to request DDoS attack mitigation assistance and receive mitigation status updates from systems of this nature.

The use cases presented in the document are intended to provide examples of communications interactions DOTS-enabled nodes in both inter- and intra-organizational DDoS mitigation scenarios. These use cases are expected to provide inputs for the design of the DOTS protocol(s).

2. Terminology and Acronyms

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Acronyms

This document makes use of the same terminology and definitions as [I-D.ietf-dots-requirements], except where noted.

2.3. Terms

Inter-organizational: a DOTS communications relationship between distinct organizations with separate spans of administrative control. Typical inter-organizational DOTS communication relationships would be between a DDoS mitigation service provider and an end-customer organizational which requires DDoS mitigation assistance; between multiple DDoS mitigation service providers coordinating mutual defense of a mutual end-customer; or between DDoS mitigation service providers which are requesting additional DDoS mitigation assistance in for attacks which exceed their inherent DDoS mitigation capacities and/or capabilities.

Intra-organizational: a DOTS communications relationship between various elements within a single span of administrative control. A typical intra-organizational DOTS communications relationship would be between DOTS clients, DOTS gateways, and DOTS servers within the same organization.

3. Use Cases Scenarios

This section provides a high-level description of scenarios addressed by DOTS. These scenarios are described in more detail in Appendix A. In both sections, the scenarios are provided in order to illustrate the use of DOTS in typical DDoS attack scenarios. They are not definitive, and other use cases are expected to emerge with widespread DOTS deployment.

All scenarios present a coordination between the targeted organization, the DDoS attack telemetry and the mitigator. The coordination and communication between these entity depends, for example on the characteristic or functionality of the equipment, the reliability of the information provided by DDoS attack telemetry, and the business relationship between the DDoS target domain and the mitigator.

More explicitly, in some cases, the DDoS telemetry attack may simply activate a DDoS mitigation, whereas in other cases, it may collaborate by providing some information about an attack. In some cases, the DDoS mitigation may be orchestrated, which includes selecting a specific appliance as well as starting/ending a mitigation.

3.1. Elementary Intra-organizational DDoS Mitigation

The most elementary scenario considers equipment such as a CPE that when overloaded sends an alert to specific equipment located upstream. In many cases, these very basic devices are unlikely to diagnose whether an DDoS attack is ongoing or not and detection as well as potential mitigation is left to the upstream equipment.

In many deployments, the upstream equipment belongs to the same organization as the CPE. In such cases, it is not expected that a specific commercial contract is established between the CPE and the DDoS mitigation service. The CPE and concerned traffic is likely to be identified by the source of the alert, which also imply the mitigator is aware of the nature of the equipment as well as the architecture of the organization.

For example, the DDoS mitigation service may be equipment that is located on path or a controller that will configure the network to the traffic to be analyzed and mitigated is redirected to a dedicated vendor specific equipment or solution. The DDoS mitigation service may be activated only for the traffic associated to the CPE sending the alert or instead to the traffic associated to all CPE. Such decisions are not part of DOTS, but instead depend on the policies of the network administrator.

The DDoS mitigation service is expected to acknowledge the reception of the alert in order to avoid retransmission. This may become an issue if an ISP receives alerts from all CPEs multiple times. However, it is unlikely that in such cases the CPE will follow the status of the mitigation. Instead, as the DDoS mitigation service and the CPE belongs to the same administrative domain, it is expected that the decision of mitigating or not, as well as the decision to end an ongoing mitigation will be left to DDoS mitigation service without notice to the CPEs.

There are several merits of using DOTS signaling in an intra-organizational manner:

1. It will facilitate interoperability between DDoS solutions/ services by providing a standards-based, programmatic communications mechanism

2. It will reduce time to initiate DDoS mitigation services

The required data exchange between DOTS client and DOTS server may be equivalent to or a subset of information set of inter-organizational use cases.

3.2. Advanced/Extended Intra-Organizational DDoS Mitigation

This section considers that more specialized equipment is generating DDoS alerts. These devices are likely to provide reliable information about the ongoing attack.

Such equipment could typically be a telemetry system, or a specific targeted service such as a web server, or another type application detecting application-specific attacks.

Typically, a telemetry system may indicate classifiers of DDoS attack traffic as well indicators or qualification of the detected attack.

As the telemetry system is expected to monitor multiple aspects of the traffic, similarly when an attack is detected by the target service.

The destination of the alert is likely to receive alerts from multiple different services (DNS, HTTP, TCP, UDP, application layer specific...). Such information is likely to be trusted and considered by the mitigator to apply to the appropriated security appliance.

Note that within a single domain it is likely that the service or the telemetry system is the most accurate equipment to qualify the attack.

As a result, not providing the information is likely to re-do the analysis phase. Providing the information while sending the alert avoid re-processing the analysis. Instead the mitigator directly uses the information to redirect the traffic to the appropriated specialized appliance.

For the same reasons as the CPE, as mitigation of the DDoS Service is performed in a single administrative domain, the source of the alert may not manage the end of the mitigation service and leave such decision to the administrator of domain or the DDoS mitigation service.

3.3. Orchestrated Intra-Organizational DDoS Mitigation

This section presents a generalization of the Service/System intra-organizational scenario. Orchestration goes one step further and considers that the information carried by the alert could have some management purpose. This includes explicitly starting/ending mitigation as well as selecting a specific DDoS mitigation service. This differs from the previous case in that the source of the alert does not leave the decision on how to mitigate the attack by the mitigator. Instead the mitigator is orchestrated.

Typical example of orchestrators could be a network administrator that monitors the traffic and manually initiates a DDoS mitigation from its web portal. Orchestration may also applied automatically by an orchestrator.

3.4. Inter-Organizational DDoS Mitigation

In the case of inter-organizational mitigation, it is expected that a DDoS mitigation service provider can provide DDoS mitigation service to the targeted organization. The relationship between the two organizations is generally expected to be described into a pre-agreed contract, although ad-hoc mitigation scenarios without a pre-existing business relationship are also quite common, and DOTS is intended to work in either scenario, once the appropriate DOTS communications relationships are configured by the involved parties.

Mutual authentication between all elements in the DOTS communications chain is required in both intra- and inter-organizational scenarios.

DDoS attacks are often sourced from multiple independent networks on the Internet. The targeted organization may request DDoS mitigation services from multiple peered DOTS organizations with cooperation contract in order to mitigate a given attack.

The coordination relationship among the DOTS organizations will often be bilateral, which represents a direct peer to peer communication between each DOTS organization without the existence of a broker or orchestrator. The other case is a broker or orchestrator facilitating DDOS mitigation coordination among multiple DOTS-enabled organizations.

4. Use Cases Taxonomy

DOTS communication is a communication between a DOTS Client and a DOTS Server. A DOTS Client or DOTS Server can be hosted on different nodes which are associated to different functionalities, and thus leading to different expectations from DOTS. This section provides a classification of the DOTS Client, DOTS Servers as well as the different examples of DOTS message exchanges.

Appendix A provides more details of anticipated DOTS communications relationships, message flow, and message type examples.

4.1. DOTS Client Taxonomy

DOTS clients initiate DOTS communications in order to request DDoS mitigation assistance. This includes initiation/termination ofDDoS mitigation service as well as requesting and reporting the status and efficacy of an ongoing DDoS mitigation.

Note that this section only considers DOTS Client that are actually initiating an exchange with a DOTS Server, and nodes that simply relay DOTS messages are not considered here.

Here are the categories of DOTS Client envisioned in this document:

(a) DOTS Client alerting a DDoS attack is ongoing

  i) hosted on the target attack

  ii) hosted on a monitoring service/system

(b) DOTS Client coordinating an DDoS attack mitigation

  i) hosted on an orchestrator

  ii) hosted on administrative GUI

When an alert is raised by the node under attack, very little information is expected to be provided by DOTS Client to the DDoS mitigation service/system. More particularly telemetric information or characteristics of the attack are likely to be unreliable as the host is already overloaded, and may not have sophisticated DDoS detection/classification capabilities.

When the DOTS Client is hosted on a more sophisticated attack monitoring system, the monitoring system may raise an alert an attack is ongoing. Unlike the host under attack, the monitoring system is expected to have sufficient resource so it is not itself overload and impacted by the ongoing attack. As a result, the DOTS Client is more likely to provide additional information associated to the alert, as this information is expected to be reliable. The type of information associated may be associated to the asset to protect and eventually some information qualifying the attack. The information associated also depends on what has been agreed with DDoS mitigation service/ system. In most cases, when a DDoS attack is detected all the traffic is redirected to the DDoS mitigation procedure that has been agreed between the DDoS mitigation service/system and the entity hosting the monitoring service. In such cases, very little information is needed.

When the DOTS Client is hosted on an orchestrator, the DOTS Client contacts the DDoS mitigation service/system to initiates a DDoS mitigation. The orchestrator is responsible for setting the network to redirect the traffic to the DDoS mitigation service/system. If the DDoS mitigation service/system is not available, the orchestrator is responsible for finding an alternative. Again the orchestrator is likely to provide additional information to the DDoS mitigation service/system. For example, typical information may be the asset to protect, as well as the specific mitigation function requested.

The service is usually expected to be associated with the mitigation service, and so may not be explicitly specified. In addition, the DOTS Client is also expected to control how the DDoS mitigation is performed. More specifically, it is expected that the DOTS Client can terminate the DDoS mitigation. The DOTS Client should have sufficient information to decide how to operate next. For example, it should be able to check if the mitigation is ongoing as well as the efficiency of the mitigation.

When the DOTS client is hosted on an administrative system, the DOTS Client may be triggered by the network administrator to initiate a DDoS mitigation. In this case, the DOTS Server is likely to be an orchestrator, and all necessary information may be provided so the DDoS mitigation can be initiated. This includes, the asset to be protected, the action expected to be performed by the orchestrator, the DDoS mitigation service/system to contact...

Note that information included by DOTS Client in a request for mitigation is not limited to simple mitigation assistance requests; it can be more detailed. However, as DDoS mitigation systems are highly heterogeneous, if there is a need to provide interoperability between the vendors and DDoS mitigation services/systems, the actions provided by a DOTS Clients remains small and accepted by all services/systems. As a result here are the envisioned optional information provided by the DOTS Client.

(a) recommended asset to protect (e.g. IP, port number, DNS record, URI, et. all.). This information specifies the expected action from the DDoS mitigation service/system.

(b) optional DDoS Mitigation Contract ID: which references the contract agreed out-of-band. This information specifies the expected action from the DDoS mitigation service/system.

(c) optional Requested Service: which designates the function or service associated to the DDoS mitigation service/system. This information specifies the expected action from the DDoS mitigation service/system.

(d) optional DDoS attack information (e.g. suspected attack, telemetry): This information is expected to help the mitigation service/system to diagnose the ongoing attack.

In both cases, the DOTS Client sends a request for DDoS mitigation to the DOTS Server, and expects the DDoS mitigation service/system to mitigate the DDoS attack. The difference between sending a request for DDoS mitigation as an alert or for coordinating an DDoS mitigation is that an alert is a request to completely outsource the mitigation, whereas the coordination requires additional control over the DDoS mitigation. An alert may be acknowledged by the DOTS Server to acknowledge the reception whereas during the coordination, the DOTS server may acknowledge the initiation of the DDoS mitigation.

4.2. DOTS Server Taxonomy

DOTS Servers terminate DOTS communications. The DOTS Server is typically hosted on a DDoS mitigation service/system or an intermediary node such as an orchestrator.

The DOTS Server is expected to be the entry point of a DDoS mitigation service/system. Some DOTS Clients do not expect any further interaction from the DOTS Server, once a DDoS mitigation has been requested. This is especially true for DOTS Clients hosted on the attack target. Other DOTS Clients hosted on orchestrators or DDoS mitigation service/systems are likely to expect from the DOTS Server a confirmation the system accepts the DDoS mitigation task.

These DOTS Client are also likely to expect a confirmation when DDoS mitigation service termination has been requested.

In addition, DOTS Servers are also expected to provide information related to the mitigation status when requested by the DOTS Client.

It is also expected that the DOTS Server could provide some status report of the DDoS mitigation on a push basis.

4.3. DOTS Message Taxonomy

The core essential messages to coordination a heterogeneous set of DDoS mitigation services/system needs to be small and enable future options. Here are the different exchanges envisioned in this document between a DOTS Client and a DOTS Server.

(a) DOTS MITIGATION CONTROL messages are used by the DOTS Client to initiate or terminate a DDoS mitigation. The initiator the termination can be specified by the action type START or STOP. These messages can carry some additional options that specify information such as the asset under attack. These DOTS MITIGATION CONTROL messages are expected to be ACKed by the DOTS Server, in order to indicate the DOTS Server will perform the requested action. In any other case an error is expected to be returned. In the case of a DOTS Client sends an alert, ACK is recommended so the DOTS Client stop sending the alert.

(b) DOTS MITIGATION INFORMATIONAL message are left for any additional interaction between a DOTS Client and DOTS Server regarding an ongoing request. An INFORMATIONAL message can be ignored by the receiver if it does not understand the requested information or options. In the current document an informational message can be the status of the ongoing mitigation.

(c) DOTS ERROR contains the errors associated to a request.

(d) DOTS OPTIONS: options can be used to indicate some optional information. The option is expected to specify whether the DOTS Server can ignore it or must return an error if it is not understood. Options are not messages, but part of the message.

5. Security Considerations

DOTS is at risk from three primary attacks: DOTS agent impersonation, traffic injection, and signaling blocking. The DOTS protocol MUST be designed for minimal data transfer to address the blocking risk.

Impersonation and traffic injection mitigation can be managed through current secure communications best practices. DOTS is not subject to anything new in this area. One consideration could be to minimize the security technologies in use at any one time. The more needed, the greater the risk of failures coming from assumptions on one technology providing protection that it does not in the presence of another technology.

Additional details of DOTS security requirements may be found in [I-D.ietf-dots-requirements].

6. IANA Considerations

No IANA considerations exist for this document at this time.

7. Acknowledgments

TBD

8. References

8.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[APACHE] "Apache mod_security", <https://www.modsecurity.org>.

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-03 (work in progress), October 2016.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RRL] "BIND RRL", <https://deepthought.isc.org/article/AA-00994/0/Using-the-Response-Rate-Limiting-Feature-in-BIND-9.10.html>.

Appendix A. Use Cases

This section provides a high-level overview of likely use cases and deployment scenarios for DOTS-enabled DDoS mitigation services. It should be noted that DOTS servers may be standalone entities which, upon receiving a DOTS mitigation service request from a DOTS client, proceed to initiate DDoS mitigation service by communicating directly or indirectly with DDoS mitigators, and likewise terminate the service upon receipt of a DOTS service termination request; conversely, the DDoS mitigators themselves may incorporate DOTS servers and/or DOTS clients. The mechanisms by which DOTS servers initiate and terminate DDoS mitigation service with DDoS mitigators is beyond the scope of this document.

All of the primary use cases described in this section are derived from current, real-world DDoS mitigation functionality, capabilities, and operational models.

The posited ancillary use cases described in this section are reasonable and highly desirable extrapolations of the functionality of baseline DOTS capabilities, and are readily attainable in the near term.

Each of the primary and ancillary use cases described in this section may be read as involving one or more DDoS mitigation service providers; DOTS makes multi-provider coordinated DDoS defenses much more effective and practical due to abstraction of the particulars of a given DDoS mitigation service/solution set.

Both the primary and ancillary use cases may be facilitated by direct DOTS client - DOTS server communications or via DOTS relays deployed in order to aggregate DOTS mitigation service requests/responses, to mediate between stateless and stateful underlying transport protocols, to aggregate multiple DOTS requests and/or responses, to filter DOTS requests and/or responses via configured policy mechanisms, or some combination of these functions.

All DOTS messages exchanged between the DOTS clients and DOTS servers in these use cases may be communicated directly between DOTS clients and servers, or mediated by one or more DOTS relays residing on the network of the originating network, the network where upstream DDoS mitigation service takes place, an intervening network or networks, or some combination of the above.

DOTS is intended to apply to both inter- and Intra-organizational DDoS attack mitigation scenarios. The technical and operational requirements for inter- and Intra-organizational DOTS communications are identical. The main difference is administrative in nature; although it should be noted that provisioning challenges which are typically associated with inter-organizational DOTS communications relationships may also apply in intra-organizational deployment scenarios, based upon organizational factors. All of the same complexities surrounding authentication and authorization can apply in both contexts, including considerations such as network access policies to allow DOTS communications; DOTS transport selection (including considerations of the implications of link congestion if a stateful DOTS transport option is selected), etc. Registration of well-known ports for DOTS transports per [RFC6335] should be considered in light of these challenges.

It should also be noted that DOTS does not directly ameliorate the various administrative challenges required for successful DDoS attack mitigation. Letters of authorization, RADB updates, DNS zone delegations, alteration of network access policies, technical configurations required to facilitate network traffic diversion and re-injection, etc., are all outside the scope of DOTS. DOTS may, however, prove useful in automating the registration of DOTS clients with DOTS servers, as well as in the automatic provisioning of situationally-appropriate DDoS defenses and countermeasures. This ancillary DOTS functionality is described in Appendix A.2.

Many of the ’external’ administrative challenges associated with establishing workable DDoS attack mitigation service may be addressed by work currently in progress in the I2RS and I2NSF WGs. Interested parties may wish to consider tracking those efforts, and coordination with both I2RS and I2NSF is highly desirable.

Note that all the use-cases in this document are universal in nature. They apply equally to endpoint networks, transit backbone providers, cloud providers, broadband access providers, ASPs, CDNs, etc. They are not specific to particular business models, topological models, or application types, and are deliberately generalizable. Both networks targeted for attack as well as any adjacent or topologically distant networks involved in a given scenario may be either single- or multi-homed. In the accompanying vector illustrations incorporated into draft-ietf-dots-use-cases-01.pdf, specific business and topological models are described in order to provide context.

Likewise, both DOTS itself and the use cases described in this document are completely independent of technologies utilized for the detection, classification, traceback, and mitigation of DDoS attacks.

Flow telemetry such as NetFlow and IPFIX, direct full-packet analysis, log-file analysis, indirection manual observation, etc. can and will be enablers for detection, classification and traceback.

Intelligent DDoS mitigation systems (IDMSes), flowspec, S/RTBH, ACLs, and other network traffic manipulation tools and techniques may be used for DDoS attack mitigation. BGP, flowspec, DNS, inline deployment, and various ’NFV’ technologies may be used for network traffic diversion into mitigation centers or devices in applicable scenarios; GRE, MPLS, ’NFV’, inline deployment and other techniques may be utilized for ’cleaned’ traffic re-injection to its intended destination.

The scope, format, and content of all DOTS message types cited in this document must be codified by the DOTS WG.

The following use cases are intended to inform the DOTS requirements described in [I-D.ietf-dots-requirements].

A.1. Primary Use Cases

A.1.1. Automatic or Operator-Assisted DOTS Clients Request Upstream DDoS Mitigation Services

DOTS client can be supported on different devices or systems, like:

- CPE or PE mitigators: CPE or PE mitigators can mitigate the DDoS attack by itself, but also with DOTS client capabilities may be configured to signal to one or more DOTS servers in order to request upstream DDoS mitigation service initiation during an attack when DDoS attack volumes and/or attack characteristics exceed the capabilities of such CPE mitigators;

- CPE or PE network infrastructure elements: Refer to the network elements like routers, switches, load-balancers, firewalls, 'IPSes', etc, which have the capability to detect and classify DDoS attacks. These network elements involved are not engaged in mitigating DDoS attack traffic, instead have DOTS client capabilities to be configured to signal to one or more DOTS servers in order to request upstream DDoS mitigation service initiation during an attack;

- CPE or PE Attack Telemetry Detection/Classification Systems: These systems having DOTS client capabilities may be configured so that upon detecting and classifying a DDoS attack, they signal one or more DOTS servers in order to request upstream DDoS mitigation service initiation. These systems do not possess any inherent capability to mitigate DDoS attack traffic, and is signaling for upstream mitigation assistance;

- The DDoS targeted service/applications: A service or application which is the target of a DDoS attack and which has the capability to detect and classify DDoS attacks (i.e, Apache mod_security [APACHE], BIND RRL [RRL], etc.) as well as DOTS client functionality may be configured so that upon detecting and classifying a DDoS attack, it signals one or more DOTS servers in order to request upstream DDoS mitigation service initiation. They do not possess any inherent capability to mitigate DDoS attack traffic, and is signaling for upstream mitigation assistance.

Despite the different implementations of DOTS client, the DOTS signaling process of them are very similar. For simplicity, the abstract term 'DOTS client' is used here as a general representation for all kinds of implementation.

One or more DOTS clients may be configured to signal to one or more DOTS servers in order to request upstream DDoS mitigation service initiation during an attack. DDoS mitigation service may be terminated either automatically or manually via a DOTS mitigation service termination request initiated by the DOTS client when it has been determined that the DDoS attack has ended. The DOTS signaling process listed below applies to both intra- and inter-organizational scenarios:

(a) A DDoS attack is initiated against online properties of an organization with DOTS clients deployed.

(b) DOTS client detects, classifies, and maybe begin mitigating the DDoS attack (if it’s implemented as the DDoS mitigator).

(c) DOTS client determine to send a DOTS mitigation service initiation request (for DDoS mitigator, if their capability to mitigate the DDoS attack is insufficient) to one or more DOTS servers residing on one or more upstream transit networks, peer networks, or overlay MSSP networks, either directly or via intermediate DOTS relays residing upon the requesting organization’s network, the upstream mitigation provider’s network, or both. This DOTS mitigation service initiation request may be automatically initiated by the DOTS clients, or may be manually triggered by personnel of the requesting organization in response to an alert from the DOTS clients (the mechanism by which this process takes place is beyond the scope of this document).

(d) The DOTS servers which receive the DOTS mitigation service initiation requests determine that they have been configured to honor requests from the requesting DOTS clients, and initiate situationally-appropriate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(e) The DOTS servers transmit a DOTS service status message to the requesting DOTS clients indicating that upstream DDoS mitigation service has been initiated.

(f) While DDoS mitigation services are active, the DOTS servers regularly transmit DOTS mitigation status updates to the requesting DOTS clients.

(g) While DDoS mitigation services are active, the DOTS clients may optionally regularly transmit DOTS mitigation efficacy updates to the relevant DOTS servers.

(h) When the upstream DDoS mitigators determine that the DDoS attack has ceased, they indicate this change in status to their respective DOTS servers (the mechanism by which this process takes place is beyond the scope of this document).

(i) The DOTS servers transmit a DOTS mitigation status update to the DOTS clients indicating that the DDoS attack has ceased.

(j) The DOTS clients transmit a DOTS mitigation service termination request to the DOTS servers. This DOTS mitigation service termination request may be automatically initiated by the DOTS clients, or may be manually triggered by personnel of the requesting organization in response to an alert from the DOTS clients or a management system which monitors them (the mechanism by which this process takes place is beyond the scope of this document).

(k) The DOTS servers terminate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(l) The DOTS servers transmit a DOTS mitigation status update to the DOTS clients indicating that DDoS mitigation services have been terminated.

(m) The DOTS clients transmit a DOTS mitigation termination status acknowledgement to the DOTS servers.

A.1.2. Manual Request to Upstream Mitigator

A Web portal, or application for mobile devices such as smartphones and tablets, which has DOTS client capabilities has been configured in order to allow authorized personnel of organizations which are targeted by DDoS attacks to manually request upstream DDoS mitigation service initiation from a DOTS server. When an organization has reason to believe that it is under active attack, authorized personnel may utilize the Web portal or mobile device application to manually initiate a DOTS client mitigation request to one or more DOTS servers in order to initiate upstream DDoS mitigation services. DDoS mitigation service may be terminated manually via a DOTS mitigation service termination request through the Web portal or mobile device application when it has been determined that the DDoS attack has ended.

In this use-case, the organization targeted for attack does not possess any automated or operator-assisted mechanisms for DDoS attack detection, classification, traceback, or mitigation; the existence of an attack has been inferred manually, and the organization is requesting upstream mitigation assistance. This can theoretically be an inter- or Intra-organizational use-case, but is more typically an inter-organizational scenario.

(a) A DDoS attack is initiated against online properties of an organization have access to a Web portal or mobile device application which incorporates DOTS client functionality and can generate DOTS mitigation service requests upon demand.

(b) Authorized personnel utilize the Web portal or mobile device application to send a DOTS mitigation service initiation request to one or more upstream transit networks, peer networks, or overlay MSSP networks, either directly or via intermediate DOTS relays residing upon the requesting organization’s network, the upstream mitigation provider’s network, or both. This DOTS mitigation service initiation request is manually triggered by personnel of the requesting organization when it is judged that the organization is under DDoS attack (the mechanism by which this process takes place is beyond the scope of this document).

(c) The DOTS servers which receive the DOTS mitigation service initiation requests determine that they have been provisioned to honor requests from the Web portal or mobile device application, and initiate situationally-appropriate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(d) The DOTS servers transmit a DOTS service status message to the Web portal or mobile device application indicating that upstream DDoS mitigation service has been initiated.

(e) While DDoS mitigation services are active, the DOTS servers regularly transmit DOTS mitigation status updates to the Web portal or mobile device application.

(f) While DDoS mitigation services are active, the Web portal or mobile device application may optionally regularly transmit manually-triggered DOTS mitigation efficacy updates to the relevant DOTS servers.

(g) When the upstream DDoS mitigators determine that the DDoS attack has ceased, they indicate this change in status to their respective DOTS servers (the mechanism by which this process takes place is beyond the scope of this document).

(h) The DOTS servers transmit a DOTS mitigation status update to the Web portal or mobile device application indicating that the DDoS attack has ceased.

(i) The Web portal or mobile device application transmits a manually-triggered DOTS mitigation service termination request to the DOTS servers (the mechanism by which this process takes place is beyond the scope of this document).

(j) The DOTS servers terminate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(k) The DOTS servers transmit a DOTS mitigation status update to the Web portal or mobile device application indicating that DDoS mitigation services have been terminated.

(l) The Web portal or mobile device application transmits a DOTS mitigation termination status acknowledgement to the DOTS servers.

A.1.3. Unsuccessful Automatic or Operator-Assisted DOTS Clients Request Upstream DDoS Mitigation Services

One or more DOTS clients may be configured to signal to one or more DOTS servers in order to request upstream DDoS mitigation service initiation during an attack (for DDoS mitigators, when DDoS attack volumes and/or attack characteristics exceed the capabilities of such mitigators). DDoS mitigation service may be terminated either automatically or manually via a DOTS mitigation service termination request initiated by the DOTS client when it has been determined that the DDoS attack has ended.

This can theoretically be an inter- or Intra-organizational use-case, but is more typically an inter-organizational scenario.

(a) A DDoS attack is initiated against online properties of an organization with DOTS clients deployed.

(b) DOTS client detects, classifies, and begins mitigating the DDoS attack (if it’s implemented as the DDoS mitigator).

(c) DOTS clients determine to send a DOTS mitigation service initiation request (for DDoS mitigator, if their capability to mitigate the DDoS attack is insufficient) to one or more DOTS servers residing on one or more upstream transit networks, peer networks, or overlay MSSP networks, either directly or via intermediate DOTS relays residing upon the requesting organization’s network, the upstream mitigation provider’s network, or both. This DOTS mitigation service initiation request may be automatically initiated by the DOTS clients, or may be manually triggered by personnel of the requesting organization in response to an alert from the DOTS clients (the mechanism by which this process takes place is beyond the scope of this document).

(d) The DOTS servers which receive the DOTS mitigation service initiation requests determine that they have been configured to honor requests from the requesting DOTS clients, and attempt to initiate situationally-appropriate DDoS mitigation service on their respective networks (the mechanism by which this process takes place is beyond the scope of this document).

(e) The DDoS mitigators on the upstream network report back to the DOTS servers that they are unable to initiate DDoS mitigation service for the requesting organization due to mitigation capacity constraints, bandwidth constraints, functionality constraints, hardware casualties, or other impediments (the mechanism by which this process takes place is beyond the scope of this document).

(f) The DOTS servers transmit a DOTS service status message to the requesting DOTS clients indicating that upstream DDoS mitigation service cannot be initiated as requested.

(g) The DOTS clients may optionally regularly re-transmit DOTS mitigation status request messages to the relevant DOTS servers until acknowledgement that mitigation services have been initiated.

(h) The DOTS clients may optionally transmit a DOTS mitigation service initiation request to DOTS servers associated with a configured fallback upstream DDoS mitigation service. Multiple fallback DDoS mitigation services may optionally be configured.

(i) The process describe above cyclically continues until the DDoS mitigation service request is fulfilled; the DOTS clients determine that the DDoS attack volume has decreased to a level and/or complexity which they themselves can successfully mitigate; the DDoS attack has ceased; or manual intervention by personnel of the requesting organization has taken place.

A.2. Ancillary Use Cases

A.2.1. Auto-registration of DOTS clients with DOTS servers

An additional benefit of DOTS is that by utilizing agreed-upon authentication mechanisms, DOTS clients can automatically register for DDoS mitigation service with one or more upstream DOTS servers.

The details of such registration are beyond the scope of this document.

A.2.2. Auto-provisioning of DDoS countermeasures

The largely manual tasks associated with provisioning effective, situationally-appropriate DDoS countermeasures is a significant barrier to providing/obtaining DDoS mitigation services for both mitigation providers and mitigation recipients. Due to the 'self-descriptive' nature of DOTS registration messages and mitigation requests, the implementation and deployment of DOTS has the potential to automate countermeasure selection and configuration for DDoS mitigators. The details of such provisioning are beyond the scope of this document.

This can theoretically be an inter- or Intra-organizational use-case, but is more typically an inter-organizational scenario.

A.2.3. Informational DDoS attack notification to interested and authorized third parties

In addition to its primary role of providing a standardized, programmatic approach to the automated and/or operator-assisted request of DDoS mitigation services and providing status updates of those mitigations to requesters, DOTS may be utilized to notify security researchers, law enforcement agencies, regulatory bodies, etc. of DDoS attacks against attack targets, assuming that organizations making use of DOTS choose to share such third-party notifications, in keeping with all applicable laws, regulations, privacy and confidentiality considerations, and contractual agreements between DOTS users and said third parties.

This is an inter-organizational scenario.

Authors' Addresses

Roland Dobbins (editor)
Arbor Networks
30 Raffles Place
Level 17 Chevron House
Singapore 048622
Singapore

Email: rdobbins@arbor.net

Stefan Fouant
Corero Network Security

Email: Stefan.Fouant@corero.com

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Robert Moskowitz
Huawei
Oak Park, MI 48237
USA

Email: rgm@labs.htt-consult.com

Nik Teague
Verisign Inc
12061 Bluemont Way
Reston, VA 20190
USA

Phone: +44 791 763 5384
Email: nteague@verisign.com

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

Email: Frank.xialiang@huawei.com

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

Email: kaname@nttv6.jp

DOTS
Internet-Draft
Intended status: Informational
Expires: January 6, 2021

R. Dobbins
Arbor Networks
D. Migault
Ericsson
R. Moskowitz
HTT Consulting
N. Teague
Iron Mountain Data Centers
L. Xia
Huawei
K. Nishizuka
NTT Communications

July 05, 2020

Use cases for DDoS Open Threat Signaling
draft-ietf-dots-use-cases-25

Abstract

The DDoS Open Threat Signaling (DOTS) effort is intended to provide protocols to facilitate interoperability across disparate DDoS mitigation solutions. This document presents sample use cases which describe the interactions expected between the DOTS components as well as DOTS messaging exchanges. These use cases are meant to identify the interacting DOTS components, how they collaborate, and what are the typical information to be exchanged.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

At the time of writing, distributed denial-of-service (DDoS) attack mitigation solutions are largely based upon siloed, proprietary communications schemes with vendor lock-in as a side-effect. This can result in the configuration, provisioning, operation, and activation of these solutions being a highly manual and often time-consuming process. Additionally, coordinating multiple DDoS mitigation solutions simultaneously is fraught with both technical and process-related hurdles. This greatly increases operational complexity which, in turn, can degrade the efficacy of mitigations that are generally highly dependent on a timely reaction by the system.

The DDoS Open Threat Signaling (DOTS) effort is intended to specify protocols that facilitate interoperability between diverse DDoS mitigation solutions and ensure greater integration in term of attack detection, mitigation requests, and attack characterization patterns.

As DDoS solutions are broadly heterogeneous among vendors, the primary goal of DOTS is to provide high-level interaction amongst differing DDoS solutions, such as detecting DDoS attacks, initiating/terminating DDoS mitigation assistance, or requesting the status of a DDoS mitigation.

This document provides sample use cases that provided input for the requirements [RFC8612] and design of the DOTS protocols [RFC8782][RFC8783]. The use cases are not exhaustive and future use cases are expected to emerge as DOTS is adopted and evolves.

2. Terminology and Acronyms

This document makes use of the same terminology and definitions as [RFC8612]. In addition it uses the terms defined below:

- DDoS Mitigation System (DMS): A system that performs DDoS mitigation. The DDoS Mitigation System may be composed of a cluster of hardware and/or software resources, but could also involve an orchestrator that may take decisions such as outsourcing some or all of the mitigation to another DDoS Mitigation System.

- DDoS Mitigation: The action performed by the DDoS Mitigation System.

- DDoS Mitigation Service: designates a service provided to a customer to mitigate DDoS attacks. Each service subscription usually involve Service Level Agreement (SLA) that has to be met. It is the responsibility of the DDoS Service provider to instantiate the DDoS Mitigation System to meet these SLAs.

- DDoS Mitigation Service Provider: designates the administrative entity providing the DDoS Mitigation Service.

- Internet Transit Provider (ITP): designates the entity that delivers the traffic to a customer network. It can be an Internet Service Provider (ISP), or an upstream entity delivering the traffic to the ISP.

3. Use Cases

3.1. Upstream DDoS Mitigation by an Upstream Internet Transit Provider

This use case describes how an enterprise or a residential customer network may take advantage of a pre-existing relation with its ITP in order to mitigate a DDoS attack targeting its network.

For clarity of discussion, the targeted network is indicated as an enterprise network, but the same scenario applies to any downstream network, including residential and cloud hosting networks.

As the ITP provides connectivity to the enterprise network, it is already on the path of the inbound and outbound traffic of the enterprise network and well aware of the networking parameters associated to the enterprise network WAN connectivity. This eases both the configuration and the instantiation of a DDoS Mitigation Service.

This section considers two kinds of DDoS Mitigation Service between an enterprise network and an ITP:

- The upstream ITP may instantiate a DDoS Mitigation System (DMS) upon receiving a request from the enterprise network. This typically corresponds to the case when the enterprise network is under attack.

- On the other hand, the ITP may identify an enterprise network as the source of an attack and send a mitigation request to the enterprise DMS to mitigate this at the source.

The two scenarios, though different, have similar interactions between the DOTS client and server. For the sake of simplicity, only the first scenario will be detailed in this section. Nevertheless, the second scenario is also in scope for DOTS.

In the first scenario, as depicted in Figure 1, an enterprise network with self-hosted Internet-facing properties such as Web servers, authoritative DNS servers, and VoIP servers has a DMS deployed to protect those servers and applications from DDoS attacks. In addition to on-premise DDoS defense capability, the enterprise has contracted with its ITP for DDoS Mitigation Services when attacks threaten to overwhelm the bandwidth of their WAN link(s).

```
       +------------------+        +------------------+
       | Enterprise       |        | Upstream         |
       | Network          |        | Internet Transit |
       |                  |        | Provider         |
       |      +--------+  |        |             DDoS Attack
       |      | DDoS   |  | <=================================
       |      | Target |  | <=================================
       |      +--------+  |        |  +------------+  |
       |                  | +-------->| DDoS       |  |
       |                  | |      |S | Mitigation |  |
       |                  | |      |  | System     |  |
       |                  | |      |  +------------+  |
       |                  | |      |                  |
       |                  | |      |                  |
       |                  | |      |                  |
       |  +------------+  | |      |                  |
       |  | DDoS       |<---+      |                  |
       |  | Mitigation |C |        |                  |
       |  | System     |  |        |                  |
       |  +------------+  |        |                  |
       +------------------+        +------------------+

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

Figure 1: Upstream Internet Transit Provider DDoS Mitigation

The enterprise DMS is configured such that if the incoming Internet traffic volume exceeds 50% of the provisioned upstream Internet WAN link capacity, the DMS will request DDoS mitigation assistance from the upstream transit provider. More sophisticated detection means may be considered as well.

The requests to trigger, manage, and finalize a DDoS Mitigation between the enterprise DMS and the ITP is performed using DOTS. The enterprise DMS implements a DOTS client while the ITP implements a DOTS server which is integrated with their DMS in this example.

When the enterprise DMS locally detects an inbound DDoS attack targeting its resources (e.g., servers, hosts, or applications), it immediately begins a DDoS Mitigation.

During the course of the attack, the inbound traffic volume to the enterprise network exceeds the 50% threshold and the enterprise DMS escalates the DDoS mitigation. The enterprise DMS DOTS client signals to the DOTS server on the upstream ITP to initiate DDoS Mitigation. The DOTS server replies to the DOTS client that it can serve this request, and mitigation is initiated on the ITP network by the ITP DMS.

Over the course of the attack, the DOTS server of the ITP periodically informs the DOTS client on the mitigation status, statistics related to DDoS attack traffic mitigation, and related information. Once the DDoS attack has ended, or decreased to a certain level that the enterprise DMS might handle by itself, the DOTS server signals the enterprise DMS DOTS client that the attack has subsided.

The DOTS client on the enterprise DMS then requests the ITP to terminate the DDoS Mitigation. The DOTS server on the ITP receives this request and once the mitigation has ended, confirms the end of upstream DDoS Mitigation to the enterprise DMS DOTS client.

The following is an overview of the DOTS communication model for this use-case:

1. A DDoS attack is initiated against resources of a network organization (here, the enterprise) which has deployed a DOTS-capable DMS - typically a DOTS client.

2. The enterprise DMS detects, classifies, and begins the DDoS Mitigation.

3. The enterprise DMS determines that its capacity and/or capability to mitigate the DDoS attack is insufficient, and sends via its DOTS client a DOTS DDoS Mitigation request to one or more DOTS servers residing on the upstream ITP.

4. The DOTS server which receives the DOTS Mitigation request determines that it has been configured to honor requests from the requesting DOTS client, and honors the request by orchestrating its own DMS.

5. While the DDoS Mitigation is active, the DOTS server regularly transmits DOTS DDoS Mitigation status updates to the DOTS client.

6. Informed by the DOTS server status update that the attack has ended or subsided, the DOTS client transmits a DOTS DDoS Mitigation termination request to the DOTS server.

7. The DOTS server terminates DDoS Mitigation, and sends the notification to the DOTS client.

Note that communications between the enterprise DOTS client and the upstream ITP DOTS server may take place in-band within the main

Internet WAN link between the enterprise and the ITP; out-of-band via a separate, dedicated wireline network link utilized solely for DOTS signaling; or out-of-band via some other form of network connectivity such as a third-party wireless 4G network connectivity.

Note also that a DOTS client that sends a DOTS Mitigation request may be also triggered by a network admin that manually confirms the request to the upstream ITP, in which case the request may be sent from an application such as a web browser or a dedicated mobile application.

Note also that when the enterprise is multihomed and connected to multiple upstream ITPs, each ITP is only able to provide a DDoS Mitigation Service for the traffic it transits. As a result, the enterprise network may be required to coordinate the various DDoS Mitigation Services associated to each link. More multi-homing considerations are discussed in [I-D.ietf-dots-multihoming].

3.2. DDoS Mitigation by a Third Party DDoS Mitigation Service Provider

This use case differs from the previous use case described in Section 3.1 in that the DDoS Mitigation Service is not provided by an upstream ITP. In other words, as represented in Figure 2, the traffic is not forwarded through the DDoS Mitigation Service Provider by default. In order to steer the traffic to the DDoS Mitigation Service Provider, some network configuration changes are required. As such, this use case is likely to apply to large enterprises or large data centers, but as for the other use cases is not exclusively limited to them.

Another typical scenario for this use case is for there to be a relationship between DDoS Mitigation Service Providers, forming an overlay of DMS. When a DDoS Mitigation Service Provider mitigating a DDoS attack reaches its resources capacity, it may chose to delegate the DDoS Mitigation to another DDoS Mitigation Service Provider.

```
      +------------------+        +------------------+
      | Enterprise       |        | Upstream         |
      | Network          |        | Internet Transit |
      |                  |        | Provider         |
      |      +--------+  |        |             DDoS Attack
      |      | DDoS   |  | <=================================
      |      | Target |  | <=================================
      |      +--------+  |        |                  |
      |                  |        |                  |
      |                  |        +------------------+
      |                  |
      |                  |        +------------------+
      |                  |        | DDoS Mitigation  |
      |                  |        | Service Provider |
      |                  |        |                  |
      |  +------------+  |        |  +------------+  |
      |  | DDoS       |<------------>| DDoS       |  |
      |  | Mitigation |C |        | S| Mitigation |  |
      |  | System     |  |        |  | System     |  |
      |  +------------+  |        |  +------------+  |
      +------------------+        +------------------+

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

Figure 2: DDoS Mitigation between an Enterprise Network and Third Party DDoS Mitigation Service Provider

In this scenario, an enterprise network has entered into a pre-arranged DDoS mitigation assistance agreement with one or more third-party DDoS Mitigation Service Providers in order to ensure that sufficient DDoS mitigation capacity and/or capabilities may be activated in the event that a given DDoS attack threatens to overwhelm the ability of the enterprise's or any other given DMS to mitigate the attack on its own.

The pre-arrangement typically includes agreement on the mechanisms used to redirect the traffic to the DDoS Mitigation Service Provider, as well as the mechanism to re-inject the traffic back to the Enterprise Network. Redirection to the DDoS Mitigation Service Provider typically involves BGP prefix announcement or DNS redirection, while re-injection of the scrubbed traffic to the enterprise network may be performed via tunneling mechanisms (e.g., GRE). The exact mechanisms used for traffic steering are out of scope of DOTS, but will need to be pre-arranged, while in some contexts such changes could be detected and considered as an attack.

In some cases the communication between the enterprise DOTS client and the DOTS server of the DDoS Mitigation Service Provider may go through the ITP carrying the DDoS attack, which would affect the communication. On the other hand, the communication between the DOTS client and DOTS server may take a path that is not undergoing a DDoS attack.

```
  +------------------+        +------------------+
  | Enterprise       |        | Upstream         |
  | Network          |        | Internet Transit |
  |                  |        | Provider         |
  |      +--------+  |        |             DDoS Attack
  |      | DDoS   |  |<----------------+         | ++====
  |      | Target |  |    Mitigated    |         | || ++=
  |      +--------+  |        |        |         | || ||
  |                  |        |        |         | || ||
  |                  |        +--------|---------+ || ||
  |                  |                 |           || ||
  |                  |        +--------|---------+ || ||
  |                  |        | DDoS Mitigation  | || ||
  |                  |        | Service Provider | || ||
  |                  |        |        |         | || ||
  |  +------------+  |        |  +------------+  | || ||
  |  | DDoS       |<------------>| DDoS       |  | || ||
  |  | mitigation |C |        |S | mitigation |<===++ ||
  |  | system     |  |        |  | system     |<======++
  |  +------------+  |        |  +------------+  |
  +------------------+        +------------------+

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

Figure 3: Redirection to a DDoS Mitigation Service Provider

When the enterprise network is under attack or at least is reaching its capacity or ability to mitigate a given DDoS attack, the DOTS client sends a DOTS request to the DDoS Mitigation Service Provider to initiate network traffic diversion - as represented in Figure 3 - and DDoS mitigation activities. Ongoing attack and mitigation status messages may be passed between the enterprise network and the DDoS Mitigation Service Provider using DOTS. If the DDoS attack has stopped or the severity of the attack has subsided, the DOTS client can request the DDoS Mitigation Service Provider to terminate the DDoS Mitigation.

3.3. DDoS Orchestration

In this use case, one or more DDoS telemetry systems or monitoring devices monitor a network - typically an ISP network, an enterprise network, or a data center. Upon detection of a DDoS attack, these DDoS telemetry systems alert an orchestrator in charge of coordinating the various DMS's within the domain. The DDoS telemetry systems may be configured to provide required information, such as a preliminary analysis of the observation, to the orchestrator.

The orchestrator analyses the various sets of information it receives from DDoS telemetry systems, and initiates one or more DDoS mitigation strategies. For example, the orchestrator could select the DDoS mitigation system in the enterprise network or one provided by the ITP.

DDoS Mitigation System selection and DDoS Mitigation techniques may depend on the type of the DDoS attack. In some case, a manual confirmation or selection may also be required to choose a proposed strategy to initiate a DDoS Mitigation. The DDoS Mitigation may consist of multiple steps such as configuring the network, or of updating already instantiated DDoS mitigation functions. Eventually, the coordination of the mitigation may involve external DDoS mitigation resources such as a transit provider or a Third Party DDoS Mitigation Service Provider.

The communication used to trigger a DDoS Mitigation between the DDoS telemetry and monitoring systems and the orchestrator is performed using DOTS. The DDoS telemetry system implements a DOTS client while the orchestrator implements a DOTS server.

The communication between a network administrator and the orchestrator is also performed using DOTS. The network administrator uses, for example, a web interface which interacts with a DOTS client, while the orchestrator implements a DOTS server.

The communication between the orchestrator and the DDoS Mitigation Systems is performed using DOTS. The orchestrator implements a DOTS client while the DDoS Mitigation Systems implement a DOTS server.

The configuration aspects of each DDoS Mitigation System, as well as the instantiations of DDoS mitigation functions or network configuration is not part of DOTS. Similarly, the discovery of available DDoS mitigation functions is not part of DOTS; and as such is out of scope.

```
          +----------+
          | network  |C           (Enterprise Network)
          | adminis  |<-+
          | trator   |  |
          +----------+  |
                        |
          +----------+  | S+--------------+     +-----------+
          |telemetry/|  +->|              |C   S| DDoS      |+
          |monitoring|<--->| Orchestrator |<--->| mitigation||
          |systems   |C   S|              |<-+  | systems   ||
          +----------+     +--------------+C |  +-----------+|
                                             |    +----------+
          -----------------------------------|-----------------
                                             |
                                             |
             (Internet Transit Provider)     |
                                             |  +-----------+
                                             | S| DDoS      |+
                                             +->| mitigation||
                                                | systems   ||
                                                +-----------+|
          * C is for DOTS client functionality    +----------+
          * S is for DOTS server functionality
```

Figure 4: DDoS Orchestration

The DDoS telemetry systems monitor various aspects of the network traffic and perform some measurement tasks.

These systems are configured so that when an event or some measurement indicators reach a predefined level their associated DOTS client sends a DOTS mitigation request to the orchestrator DOTS server. The DOTS mitigation request may be associated with some optional mitigation hints to let the orchestrator know what has triggered the request. In particular, it is possible for something that locally to one telemetry system looks like an attack is not actually an attack when seen from the broader scope (e.g., of the orchestrator)

Upon receipt of the DOTS mitigation request from the DDoS telemetry system, the orchestrator DOTS server responds with an acknowledgment, to avoid retransmission of the request for mitigation. The orchestrator may begin collecting additional fine-grained and specific information from various DDoS telemetry systems in order to correlate the measurements and provide an analysis of the event. Eventually, the orchestrator may ask for additional information from the DDoS telemetry system; however, the collection of this information is out of scope of DOTS.

The orchestrator may be configured to start a DDoS Mitigation upon approval from a network administrator. The analysis from the orchestrator is reported to the network administrator via, for example, a web interface. If the network administrator decides to start the mitigation, the network administrator triggers the DDoS mitigation request using, for example, a web interface of a DOTS client communicating to the orchestrator DOTS server. This request is expected to be associated with a context that provides sufficient information to the orchestrator DOTS server to infer, elaborate and coordinate the appropriate DDoS Mitigation.

Upon receiving a request to mitigate a DDoS attack aimed at a target, the orchestrator may evaluate the volume of the attack as well as the value that the target represents. The orchestrator may select the DDoS Mitigation Service Provider based on the attack severity. It may also coordinate the DDoS Mitigation performed by the DDoS Mitigation Service Provider with some other tasks such as, for example, moving the target to another network so new sessions will not be impacted. The orchestrator requests a DDoS Mitigation by the selected DDoS mitigation systems via its DOTS client, as described in Section 3.1.

The orchestrator DOTS client is notified that the DDoS Mitigation is effective by the selected DDoS mitigation systems. The orchestrator DOTS server returns this information back to the network administrator.

Similarly, when the DDoS attack has stopped, the orchestrator DOTS client is notified and the orchestrator’s DOTS server indicates to the DDoS telemetry systems as well as to the network administrator the end of the DDoS Mitigation.

In addition to the above DDoS Orchestration, the selected DDoS mitigation system can return back a mitigation request to the orchestrator as an offloading. For example, when the DDoS attack becomes severe and the DDoS mitigation system’s utilization rate reaches its maximum capacity, the DDoS mitigation system can send mitigation requests with additional hints such as its blocked traffic information to the orchestrator. Then the orchestrator can take further actions such as requesting forwarding nodes such as routers to filter the traffic. In this case, the DDoS mitigation system implements a DOTS client while the orchestrator implements a DOTS server. Similar to other DOTS use cases, the offloading scenario assumes that some validation checks are followed by the DMS, the orchestrator, or both (e.g., avoid exhausting the resources of the forwarding nodes or inadvertent disruption of legitimate services). These validation checks are part of the mitigation, and are therefore out of the scope of the document.

4. Security Considerations

The document does not describe any protocol, though there are still a few high-level security considerations to discuss.

DOTS is at risk from three primary attacks: DOTS agent impersonation, traffic injection, and signaling blocking.

Impersonation and traffic injection mitigation can be mitigated through current secure communications best practices including mutual authentication. Preconfigured mitigation steps to take on the loss of keepalive traffic can partially mitigate signal blocking, but in general it is impossible to comprehensively defend against an attacker that can selectively block any or all traffic. Alternate communication paths that are (hopefully) not subject to blocking by the attacker in question is another potential mitigation.

Additional details of DOTS security requirements can be found in [RFC8612].

Service disruption may be experienced if inadequate mitigation actions are applied. These considerations are out of the scope of DOTS.

5. IANA Considerations

No IANA considerations exist for this document.

6. Acknowledgments

The authors would like to thank among others Tirumaleswar Reddy; Andrew Mortensen; Mohamed Boucadair; Artyom Gavrichenkov; Jon Shallow, Yuuhei Hayashi, Elwyn Davies, the DOTS WG chairs, Roman Danyliw and Tobias Gondrom as well as the Security AD Benjamin Kaduk for their valuable feedback.

We also would like to thank Stephan Fouant that was part of the initial co-authors of the documents.

7. Informative References

[I-D.ietf-dots-multihoming]
Boucadair, M., Reddy.K, T., and W. Pan, "Multi-homing Deployment Considerations for Distributed-Denial-of-Service Open Threat Signaling (DOTS)", draft-ietf-dots-multihoming-04 (work in progress), May 2020.

[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>.

[RFC8782] Reddy.K, T., Ed., Boucadair, M., Ed., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 8782, DOI 10.17487/RFC8782, May 2020, <https://www.rfc-editor.org/info/rfc8782>.

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

Authors' Addresses

Roland Dobbins
Arbor Networks
Singapore

EMail: rdobbins@arbor.net

Daniel Migault
Ericsson
8275 Trans Canada Route
Saint Laurent, QC 4S 0B6
Canada

EMail: daniel.migault@ericsson.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
USA

EMail: rgm@labs.htt-consult.com

Nik Teague
Iron Mountain Data Centers
UK

EMail: nteague@ironmountain.co.uk

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

EMail: Frank.xialiang@huawei.com

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

EMail: kaname@nttv6.jp

DOTS
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2017

R. Moskowitz
L. Xia
Huawei
D. Migault
Ericsson
A. Mortensen
Arbor Networks, Inc.

October 30, 2016

Strong Identities for DOTS Agents
draft-moskowitz-dots-identities-00.txt

Abstract

DOTS communications are machine-to-machine oriented communications. In addition DOTS agents are expected to end up in a large number of entities. As a result, in addition to secure, the naming scheme to identify all DOTS agents must be scalable. For these reasons this document recommends the use of cryptographic identifiers or strong Identities as opposed to human readable identifiers for example.

This document proposes two forms of strong Identities for the registration and operation of DOTS Agents. One is 802.1AR LDevID [Std-802.1AR-2009] X.509 certificates. The other is raw public keys as in HIP [RFC7401] or TLS/DTLS Raw Public Keys [RFC7250].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DOTS communications are machine-to-machine oriented communications. In addition DOTS agents are expected to end up in a large number of entities. As a result, in addition to secure, the naming scheme to identify all DOTS agents must be scalable. For these reasons the document recommend the use of cryptographic identifiers or strong Identities as opposed to human readable identifiers for example.

Human readable identifiers are very helpful to represent a resource for human. A typical example is the use of First Name and Last Name which is easier for human beings to remember than a phone number. The same occurs for web sites where FQDNs are easier to remember than the IPv6 addresses. However the human readable representation also comes with some issues.

First human readable identifiers have very little entropy which means that when the number of identifiers grow, collision are likely to happen. The likelihood of identifier collision may be limited at the expense of management complexity whose complexity grows with the number of identifiers. Second, human readable identifiers are only meaningful when used by humans who are only able to handle a limited numbers of identifiers. Third the identifier needs to be securely bound to additional element such as security elements - a public key - or routing elements - an IP address - in order to enable a communication.

As a result, human readable identifiers do not scale to meet the need of DOTs identifiers and the management overhead complexity to make identifiers human readable becomes meaningless in a automated machine to machine environment. A DOTS is intended for machine to machine -like communication, there is no reason for using human readable identifiers. DOTS recommends the use of cryptographic identifiers to avoid an additional and unnecessary cryptographic binding between the identifier and the security material.

This document describes two forms of strong Identities for the registration and operation of DOTS Agents.

1.1. X.509 LDevID

The first is the X.509 LDevID defined in 802.1AR [Std-802.1AR-2009]. The methodology proposed herein expects the DOTS mitigation provider to provide a PKI that will issue LDevID certificates to its customers. This provides a strong "domain of trust" to the identities of its customers. Inter provider trust can be established through any of the multi-PKI trust models in use today.

Customer LDevID registration may be based on an "Owner Certificate", allocated to the device during a NETCONF zerotouch registration [I-D.ietf-netconf-zerotouch]. Or the IDevID could be used directly in some registration process.

1.2. Raw Public Key

The second form of Identity is a Raw Public Key.

One type of Raw Public Key is a HIP [RFC7401] Host Identity (HI). The customer may use HIP DNS Extension [RFC8005] to assert its HI to the DOTS mitigation provider and then use HIP to prove ownership of the HI.

Although nothing prevents HI/HIT to be assigned by the provider, there is currently no mechanisms defined for such provisioning. This might be defined in future work, however, the current use of HI/HIT is that these identifiers are generated by the owner or the agent.

Another type of Raw Public Key is defined in [RFC7250]. The customer would use the methods defined in DANE [RFC6698] validate a TLS Raw Public Key.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Definitions

DOTS Agents: Per [I-D.ietf-dots-requirements], any DOTS-aware software module capable of participating in a DOTS signaling session.

Host Identity (HI): The term "HI" is defined in [RFC7401] as "the public key of the signature algorithm that represents the identity of the host." In HIP, a host proves its identity by creating a signature with the private key belonging to its HI.

Initial Secure Device Identifier (IDevID): The term "IDevID" is defined in [Std-802.1AR-2009] as "the Secure Device Identifier (DevID) installed on the device by the manufacturer". By example, an IDevID certificate, signed by the manufacturer may encode a manufacturer assigned unique identifier (e.g., serial number) and a public key matching a private key held within a TPM chip embedded within the device.

Locally Significant Secure Device Identifier (LDevID): The term "LDevID" is defined in [Std-802.1AR-2009] as "A Secure Device Identifier credential that is unique in the local administrative domain in which the device is used.". By example, an LDevID certificate, signed by the device owner may encode an owner assigned unique identifier (e.g., installation location) and a public key matching a private key held within a TPM chip embedded within the device.

Owner Certificate: The term "owner certificate" is defined in [I-D.ietf-netconf-zerotouch] as used in this document to represent an X.509 certificate, signed by the device's manufacturer or delegate, that binds an owner identity to the owner's private key, which the owner can subsequently use to sign artifacts.

TLS Raw Public Key: The term "Raw Public Key" is defined in [RFC7250]. So a "TLS Raw Public Key" as used in this document to represent this subset of an X.509 certificate used in the manner specified in RFC7250.

3. Problem Space

3.1. Trusted Identities

DOTS is meant to be deployed within the context of a business arrangement between a customer and a DDoS mitigation provider. This relationship is long-lived over a persistent session. This relationship and the data communications can best be managed with trusted identities.

Further, the customer's DDoS mitigation provider may need to enlist the assistance of a peer provider. A strong trusted identity link from the requesting provider back to the attacked customer would benefit the mitigation process.

3.2. Managing the scope of Trust

The web's PKI model is fraught with trust leakage challenges. Why trust a specific certificate just because some CA within the list of CAs that must be trusted has signed the certificate? This leads to independent vetting of each client certificate or applying some rules as to which CA is accepted for what business arrangement and then still maintaining a list of accepted client certificates is needed. This leads to a basic question of what is an X.509 certificate providing to the business agreement that would not be needed to be found out independent from the certificate?

4. Effectively Managing Identity Trust

4.1. The IEEE 802.1AR Device Identity Certificate Model

IEEE 802.1AR [Std-802.1AR-2009] defines two important types of X.509 certificates. The IDevID is installed in the device in permanent, secure storage (e.g. a TPM) and is NEVER replaced. This certificate is signed by the manufacturer's CA. Typically, its subjectName contains the device's serial number and other information that uniquely identifies the device.

The IDevID is not appropriate to use as an Identifier for any action other than provisioning another Identifier that is more flexible for general use. This limitation is based, in part, on the permanency of IDevIDs and the potential for a large number of CAs 'owning' those IDevIDs.

The second, more regularly usable, type of certificate is the LDevID. This Locally Significant Secure Device Identifier is expected to be signed by a PKI appropriate to its use. For example, the DDoS mitigation provider can maintain its PKI for the signing and validating the device's LDevID certificate. The methodology for an IDevID to leverage the creation of an LDevID is left to IETF protocols. Originally, this meant using PKIX protocols like CMP [RFC4210]. Recent work with [I-D.ietf-netconf-zerotouch] can lead to a trusted lDevID request based on the Owner Certificate.

4.2. The Raw Public Key Model

With Raw Public Keys, the trust establishment is left to the provider. Authentication based on a Raw Public Key assumes the peer already has the corresponding identifier. Authentication based on raw keys has been integrated by many protocol such as IKEv2, HIP, and TLS. However, unless the identifier is known by the peer, such protocol end with an unauthenticated communication. Provisioning of the identifier, is usually out of scope of these protocol. The Identifier may be provided out of band, using leap of faith mechanisms. Eventually DNSSEC can also be used to bind the identifier to raw key.

There are some recent developments, like Hierarchical HITs [I-D.moskowitz-hierarchical-hip] that provide an trusted infrastructure for Raw Public Keys.

5. IANA Considerations

TBD

6. Security Considerations

TBD

7. Acknowledgments

TBD

8. References

8.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[Std-802.1AR-2009] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

8.2. Informative References

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-02 (work in progress), July 2016.

[I-D.ietf-netconf-zerotouch] Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-09 (work in progress), July 2016.

[I-D.moskowitz-hierarchical-hip] Moskowitz, R. and X. Xu, "Hierarchical HITs for HIPv2", draft-moskowitz-hierarchical-hip-02 (work in progress), October 2016.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <http://www.rfc-editor.org/info/rfc4210>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[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, <http://www.rfc-editor.org/info/rfc7250>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

[RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <http://www.rfc-editor.org/info/rfc8005>.

Authors’ Addresses

Robert Moskowitz
Huawei
Oak Park, MI 48237
USA

Email: rgm@labs.htt-consult.com

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

Email: Frank.xialiang@huawei.com

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

Email: amortensen@arbor.net

DOTS   T. Reddy
Internet-Draft   Cisco
Intended status: Standards Track   M. Boucadair
Expires: May 20, 2017   Orange
                                                                P. Patil
                                                                   Cisco
                                                                 D. Wing
                                                       November 16, 2016

Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel
draft-reddy-dots-signal-channel-04

Abstract

This document specifies a mechanism that a DOTS client can use to signal that a network is under a Distributed Denial-of-Service (DDoS) attack to an upstream DOTS server so that appropriate mitigation actions are undertaken (including, blackhole, drop, rate-limit, or add to watch list) on the suspect traffic. The document specifies the DOTS signal channel including Happy Eyeballs considerations. The specification of the DOTS data channel is elaborated in a companion document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A distributed denial-of-service (DDoS) attack is an attempt to make machines or network resources unavailable to their intended users. In most cases, sufficient scale 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.

In many cases, it may not be possible for an enterprise network administrators to determine the causes of an attack, but instead just realize that certain resources seem to be under attack. This document, which adheres to the DOTS architecture [I-D.ietf-dots-architecture], proposes that, in such cases, the DOTS client just inform its DOTS server(s) that the enterprise is under a potential attack and that the mitigator monitor traffic to the enterprise to mitigate any possible attacks. This cooperation between DOTS agents contributes to ensure a highly automated network that is also robust, reliable and secure.

Protocol requirements for DOTS signal channel are obtained from DOTS requirements [I-D.ietf-dots-requirements].

This document satisfies all the use cases discussed in [I-D.ietf-dots-use-cases] except the Third-party DOTS notifications use case in Section 3.2.3 of [I-D.ietf-dots-use-cases] which is an optional feature and not a core use case. Third-party DOTS notifications are not part of the DOTS requirements document. Moreover, the DOTS architecture does not assess whether that use case may have an impact on the architecture itself and/or the DOTS trust model.

This is a companion document to the DOTS data channel specification [I-D.reddy-dots-data-channel].

2. Notational Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

(D)TLS: For brevity this term is used for statements that apply to both Transport Layer Security [RFC5246] and Datagram Transport Layer Security [RFC6347]. Specific terms will be used for any statement that applies to either protocol alone.

The reader should be familiar with the terms defined in [I-D.ietf-dots-architecture].

3. Solution Overview

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

TCP DDoS SYN-flood, for example, is a memory-exhaustion attack on the victim and ACK-flood is a CPU exhaustion attack on the victim ([RFC4987]). Attacks on the link are carried out by sending enough traffic such that the link becomes excessively congested, and legitimate traffic suffers high packet loss. Stateful firewalls can also be attacked by sending traffic that causes the firewall to hold excessive state and the firewall runs out of memory, and can no longer instantiate the state required to pass legitimate flows. Other possible DDoS attacks are discussed in [RFC4732].

In each of the cases described above, the possible arrangements between the DOTS client and DOTS server to mitigate the attack are discussed in [I-D.ietf-dots-use-cases]. An example of network diagram showing a deployment of these elements is shown in Figure 1. Architectural relationships between involved DOTS agents is explained in [I-D.ietf-dots-architecture]. In this example, the DOTS server is operating on the access network.

```
   Network
   Resource        CPE router            Access network     __________
 +-----------+    +--------------+       +-------------+    /          \
 |           |____|              |_______|             |___ | Internet |
 |DOTS client|    | DOTS gateway |       | DOTS server |    |          |
 |           |    |              |       |             |    |          |
 +-----------+    +--------------+       +-------------+    \__________/
```

Figure 1

The DOTS server can also be running on the Internet, as depicted in Figure 2.

```
   Network                                               DDoS mitigation
   Resource        CPE router             __________         service
  +-----------+    +-------------+       /          \    +-------------+
  |           |____|             |_______|          |___ |             |
  |DOTS client|    |DOTS gateway |       | Internet |    | DOTS server |
  |           |    |          |  |       |          |    |             |
  +-----------+    +-------------+       \__________/    +-------------+
```

Figure 2

In typical deployments, the DOTS client belongs to a different administrative domain than the DOTS server. For example, the DOTS client is a web server serving content owned and operated by an domain, while the DOTS server is owned and operated by a different domain providing DDoS mitigation services. That domain providing DDoS mitigation service might, or might not, also provide Internet access service to the website operator.

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 the DOTS server or indirectly via a DOTS gateway.

This document focuses on the DOTS signal channel.

4. Happy Eyeballs for DOTS Signal Channel

DOTS signaling can happen with DTLS [RFC6347] over UDP and TLS [RFC5246] over TCP. A DOTS client can use DNS to determine the IP address(es) of a DOTS server or a DOTS client may be provided with the list of DOTS server IP addresses. The DOTS client MUST know a DOTS server’s domain name; hard-coding the domain name of the DOTS server into software is NOT RECOMMENDED in case the domain name is not valid or needs to change for legal or other reasons. The DOTS client performs A and/or AAAA record lookup of the domain name and the result will be a list of IP addresses, each of which can be used to contact the DOTS server using UDP and TCP.

If an IPv4 path to reach a DOTS server is found, but the DOTS server’s IPv6 path is not working, a dual-stack DOTS client can experience a significant connection delay compared to an IPv4-only DOTS client. The other problem is that if a middlebox between the DOTS client and DOTS server is configured to block UDP, the DOTS client will fail to establish a DTLS session with the DOTS server and will, then, have to fall back to TLS over TCP incurring significant connection delays. [I-D.ietf-dots-requirements] discusses that DOTS client and server will have to support both connectionless and connection-oriented protocols.

To overcome these connection setup problems, the DOTS client can try connecting to the DOTS server using both IPv6 and IPv4, and try both DTLS over UDP and TLS over TCP in a fashion similar to the Happy Eyeballs mechanism [RFC6555]. These connection attempts are performed by the DOTS client when its initializes, and the client uses that information for its subsequent alert to the DOTS server. In order of preference (most preferred first), it is UDP over IPv6, UDP over IPv4, TCP over IPv6, and finally TCP over IPv4, which adheres to address preference order [RFC6724] and the DOTS preference that UDP be used over TCP (to avoid TCP’s head of line blocking).

```
DOTS client                                               DOTS server
   |                                                         |
   |--DTLS ClientHello, IPv6 ---->X                          |
   |--TCP SYN, IPv6-------------->X                          |
   |--DTLS ClientHello, IPv4 ---->X                          |
   |--TCP SYN, IPv4----------------------------------------->|
   |--DTLS ClientHello, IPv6 ---->X                          |
   |--TCP SYN, IPv6-------------->X                          |
   |<-TCP SYNACK---------------------------------------------|
   |--DTLS ClientHello, IPv4 ---->X                          |
   |--TCP ACK----------------------------------------------->|
   |<------------Establish TLS Session---------------------->|
   |----------------DOTS signal----------------------------->|
   |                                                         |
```

Figure 3: Happy Eyeballs

In reference to Figure 3, the DOTS client sends two TCP SYNs and two DTLS ClientHello messages at the same time over IPv6 and IPv4. In this example, it is assumed that the IPv6 path is broken and UDP is dropped by a middle box but has little impact to the DOTS client because there is no long delay before using IPv4 and TCP. The IPv6 path and UDP over IPv6 and IPv4 is retried until the DOTS client gives up.

5. DOTS Signal Channel

5.1. Overview

Constrained Application Protocol (CoAP) [RFC7252] is used for DOTS signal channel (Figure 4). COAP was designed according to the REST architecture, and thus exhibits functionality similar to that of HTTP, it is quite straightforward to map from CoAP to HTTP and from HTTP to CoAP. CoAP has been defined to make use of both DTLS over

UDP and TLS over TCP [I-D.ietf-core-coap-tcp-tls]. The advantages of COAP are: (1) Like HTTP, CoAP is based on the successful REST model, (2) CoAP is designed to use minimal resources, (3) CoAP integrates with JSON, CBOR or any other data format, (4) asynchronous message exchanges, (5) includes a congestion control mechanism (6) allows configuration of message transmission parameters specific to the application environment (including dynamically adjusted values, see Section 4.8.1 in [RFC7252]) etc.

```
+--------------+
|    DOTS      |
+--------------+
|     CoAP     |
+--------------+
| TLS |  DTLS  |
+--------------+
| TCP |   UDP  |
+--------------+
|    IP        |
+--------------+
```

Figure 4: Abstract Layering of DOTS signal channel over CoAP over (D)TLS

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 client SHOULD re-use the (D)TLS session.

Concise Binary Object Representation (CBOR) [RFC7049] is a binary encoding designed for small code and message size, CBOR encoded payloads are used to convey signal channel specific payload messages that convey request parameters and response information such as errors.

5.2. Mitigation Service Requests

The following APIs define the means to convey a DOTS signal from a DOTS client to a DOTS server:

POST requests: are used to convey the DOTS signal from a DOTS client to a DOTS server over the signal channel, possibly traversing a DOTS gateway, indicating the DOTS client’s need for mitigation, as well as the scope of any requested mitigation (Section 5.2.1). DOTS gateway act as a CoAP-to-CoAP Proxy (explained in [RFC7252]).

DELETE requests: are used by the DOTS client to withdraw the request for mitigation from the DOTS server (Section 5.2.2).

GET requests: are used by the DOTS client to retrieve the DOTS signal(s) it had conveyed to the DOTS server (Section 5.2.3).

PUT requests: are used by the DOTS client to convey mitigation efficacy updates to the DOTS server (Section 5.2.4).

Reliability is provided to the POST, DELETE, GET, and PUT requests by marking them as Confirmable (CON) 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]. Reliability is provided to the responses by marking them as Confirmable (CON) messages. The DOTS server can either piggback the response in the acknowledgement message or if the DOTS server is not able to 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 message is explained in Section 2.2 of [RFC7252]. When the response is ready, the server sends it in a new Confirmable message which then in turn needs to be acknowledged by the DOTS client (see Sections 5.2.1 and Sections 5.2.2 in [RFC7252]).

Implementation Note: A DOTS client that receives a response in a CON 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 CON, the DOTS client may no longer have any state to which to correlate this response, making the retransmission an unexpected message; 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 in [RFC7252] for more details).

5.2.1. Convey DOTS Signals

When suffering an attack and desiring DoS/DDoS mitigation, a DOTS signal is sent by the DOTS client to the DOTS server. A POST request is used to convey a DOTS signal to the DOTS server (Figure 5). The DOTS server can enable mitigation on behalf of the DOTS client by communicating the DOTS client’s request to the mitigator and relaying any mitigator feedback to the requesting DOTS client.

```
     Header: POST (Code=0.02)
     Uri-Host: "host"
     Uri-Path: ".well-known"
     Uri-Path: "DOTS-signal"
     Uri-Path: "version"
     Content-Type: "application/cbor"
     {
        "policy-id": "integer",
        "target-ip": "string",
        "target-port": "string",
        "target-protocol": "string",
        "FQDN": "string",
        "URI": "string",
        "E.164": "string",
        "alias": "string"
        "lifetime": "integer"
      }
```

Figure 5: POST to convey DOTS signals

The header fields are described below.

policy-id: Identifier of the policy represented using an integer. This identifier MUST be unique for each policy bound to the DOTS client, i.e. ,the policy-id needs to be unique relative to the active policies with the DOTS server. This identifier MUST be generated by the DOTS client. This document does not make any assumption about how this identifier is generated. This is a mandatory attribute.

target-ip: A list of IP addresses or prefixes under attack. IP addresses and prefixes are separated by commas. Prefixes are represented using CIDR notation [RFC4632]. This is an optional attribute.

target-port: A list of ports under attack. Ports are separated by commas and port number range (using "-"). For TCP, UDP, SCTP, or DCCP: the range of ports (e.g., 1024-65535). This is an optional attribute.

target-protocol: A list of protocols under attack. Valid protocol values include tcp, udp, sctp, and dccp. Protocol values are separated by commas. This is an optional attribute.

FQDN: Fully Qualified Domain Name, is the full name of a system, rather than just its hostname. For example, "venera" is a hostname, and "venera.isi.edu" is an FQDN. This is an optional attribute.

URI: Uniform Resource Identifier (URI). This is an optional attribute.

E.164: E.164 number. This is an optional attribute.

alias: Name of the alias (see Section 3.1.1 in [I-D.reddy-dots-data-channel]). This is an optional attribute.

lifetime: Lifetime of the mitigation request policy in seconds. 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. The default lifetime of the policy is 60 minutes -- this value was chosen to be long enough so that refreshing is not typically a burden on the DOTS client, while expiring the policy where the client has unexpectedly quit in a timely manner. A lifetime of zero indicates indefinite lifetime for the mitigation request. The server MUST always indicate the actual lifetime in the response. This is an optional attribute in the request.

In the POST request at least one of the attributes target-ip or target-port or target-protocol or FQDN or URI or E.164 or alias MUST be present. The relative order of two mitigation requests is determined by comparing their respective policy identifiers. The mitigation request with higher numeric policy identifier value has higher precedence (and thus will match before) than the mitigation request with lower numeric policy identifier value.

To avoid DOTS signal message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. If the Path MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP is used to convey the DOTS signal messages then 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 overhead of 13 octets + authentication overhead of the negotiated DTLS cipher suite + block padding (Section 4.1.1.1 of [RFC6347]]. If the 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-ip’ field could be split into multiple lists and each list conveyed in a new POST 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 absolutely ensure that there is no IP fragmentation. If IPv4 support on unusual networks is a consideration and path MTU is unknown, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes, as per [RFC0791] IP packets up to 576 bytes should never need to be fragmented, thus sending a maximum of 500 bytes of DOTS signal over a UDP datagram will generally avoid IP fragmentation.

Figure 6 shows a POST request to signal that ports 80, 8080, and 443 on the servers 2002:db8:6401::1 and 2002:db8:6401::2 are being attacked (illustrated in human-readable text format).

```
  Header: POST (Code=0.02)
  Uri-Host: "www.example.com"
  Uri-Path: ".well-known"
  Uri-Path: "v1"
  Uri-Path: "DOTS-signal"
  Content-Format: "application/cbor"
  {
    "policy-id":123321333242,
    "target-ip":[
        "2002:db8:6401::1",
        "2002:db8:6401::2"
    ],
    "target-port":[
        "80",
        "8080",
        "443"
    ],
    "target-protocol":"tcp"
  }
```

The CBOR encoding format is shown below:

```
a4                                    # map(4)
   69                                 # text(9)
      706f6c6963792d6964              # "policy-id"
   1b 0000001cb68635fa                # unsigned(123321333242)
   69                                 # text(9)
      7461726765742d6970              # "target-ip"
   82                                 # array(2)
      70                              # text(16)
         323030323a6462383a363430313a3a31 # "2002:db8:6401::1"
      70                              # text(16)
         323030323a6462383a363430313a3a32 # "2002:db8:6401::2"
   6b                                 # text(11)
```

```
         7461726765742d706f7274            # "target-port"
      83                                   # array(3)
         62                                # text(2)
            3830                           # "80"
         64                                # text(4)
            38303830                       # "8080"
         63                                # text(3)
            343433                         # "443"
      6f                                   # text(15)
         7461726765742d70726f746f636f6c    # "target-protocol"
      63                                   # text(3)
         746370                            # "tcp"
```

Figure 6: POST for DOTS signal

The DOTS server indicates the result of processing the POST request using CoAP response codes. CoAP 2.xx codes are success, CoAP 4.xx codes are some sort of invalid requests and 5.xx codes are returned if the DOTS server has erred or is incapable of performing the mitigation. Response code 2.01 (Created) will be returned in the response if the DOTS server has accepted the mitigation request and will try to mitigate the attack. If the request is missing one or more mandatory attributes, then 4.00 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 4.02 (Invalid query) will be returned in the response. The CoAP response will include the CBOR body received in the request.

5.2.2. Withdraw a DOTS Signal

A DELETE request is used to withdraw a DOTS signal from a DOTS server (Figure 7).

```
  Header: DELETE (Code=0.04)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "version"
  Uri-Path: "DOTS-signal"
  Content-Format: "application/cbor"
  {
     "policy-id": "integer"
  }
```

Figure 7: Withdraw DOTS signal

If the DOTS server does not find the policy number conveyed in the DELETE request in its policy state data, then it responds with a 4.04

(Not Found) error response code. The DOTS server successfully acknowledges a DOTS client’s request to withdraw the DOTS signal using 2.02 (Deleted) response code, and ceases mitigation activity as quickly as possible.

5.2.3. Retrieving a DOTS Signal

A GET request is used to retrieve information and status of a DOTS signal from a DOTS server (Figure 8). If the DOTS server does not find the policy number conveyed in the GET request in its policy state data, then it responds with a 4.04 (Not Found) error response code.

1) To retrieve all DOTS signals signaled by the DOTS client.

```
  Header: GET (Code=0.01)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "version"
  Uri-Path: "DOTS-signal"
  Observe : 0
```

2) To retrieve a specific DOTS signal signaled by the DOTS client. The policy information in the response will be formatted in the same order it was processed at the DOTS server.

```
  Header: GET (Code=0.01)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "version"
  Uri-Path: "DOTS-signal"
  Uri-Path: "policy-id value"
  Observe : 0
```

Figure 8: GET to retrieve the rules

Figure 9 shows the response of all the active policies on the DOTS server.

```
{
  "policy-data":[
    {
      "policy-id":123321333242,
      "target-protocol":"tcp",
      "lifetime":3600,
      "status":"mitigation in progress"
    },
    {
      "policy-id":123321333244,
      "target-protocol":"udp",
      "lifetime":1800,
      "status":"mitigation complete"
    },
    {
      "policy-id":123321333245,
      "target-protocol":"tcp",
      "lifetime":1800,
      "status":"attack stopped"
    }
  ]
}
```

Figure 9: Response body

The various possible values of status field are explained below:

mitigation in progress: Attack mitigation is in progress (e.g., changing the network path to re-route the inbound traffic to DOTS mitigator).

mitigation complete: Attack is successfully mitigated (e.g., attack traffic is dropped).

attack stopped: Attack has stopped and the DOTS client can withdraw the mitigation request.

mitigation capacity exceeded: Attack has exceeded the mitigation provider capability.

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. A DOTS client conveys the observe option set to 0 in the GET request to receive unsolicited notifications of attack mitigation status from the DOTS server. Unidirectional notifications within the bidirectional signal channel allows unsolicited message delivery, enabling asynchronous notifications between the agents. A DOTS client that is no longer interested in receiving notifications from the DOTS server can simply "forget" the observation. When the DOTS server then 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.

```
                       DOTS Client            DOTS Server
                          |                           |
                          |  GET /<policy-id number>  |
                          |  Token: 0x4a              |   Registration
                          |  Observe: 0               |
                          +-------------------------->|
                          |                           |
                          |  2.05 Content             |
                          |  Token: 0x4a              |   Notification of
                          |  Observe: 12              |   the current state
                          |  status: "mitigation      |
                          |          in progress"     |
                          |<--------------------------+
                          |  2.05 Content             |
                          |  Token: 0x4a              |   Notification upon
                          |  Observe: 44              |    a state change
                          |  status: "mitigation      |
                          |          complete"        |
                          |<--------------------------+
                          |  2.05 Content             |
                          |  Token: 0x4a              |   Notification upon
                          |  Observe: 60              |   a state change
                          |  status: "attack stopped" |
                          |<--------------------------+
                          |                           |
```

Figure 10: Notifications of attack mitigation status

5.2.3.1. Mitigation Status

A DOTS client retrieves the information about a DOTS signal at frequent intervals to determine the status of an attack. 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, and the DOTS client recalls the mitigation request.

A DOTS client should react to the status of the attack from the DOTS server and not the fact that it has recognized, using its own means, that the attack has been mitigated. This ensures that the DOTS client does not recall a mitigation request in a premature fashion 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 and the attack is not completely averted.

5.2.4. Efficacy Update from DOTS Client

While DDoS mitigation is active, a DOTS client MAY frequently transmit DOTS mitigation efficacy updates to the relevant DOTS server. An PUT request (Figure 11) is used to convey the mitigation efficacy update to the DOTS server. The PUT request MUST include all the header fields used in the POST request to convey the DOTS signal (Section 5.2.1). If the DOTS server does not find the policy number conveyed in the PUT request in its policy state data, it responds with a 4.04 (Not Found) error response code.

```
Header: PUT (Code=0.03)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "policy-id value"
Content-Format: "application/cbor"
{
  "target-ip": "string",
  "target-port": "string",
  "target-protocol": "string",
  "FQDN": "string",
  "URI": "string",
  "E.164": "string",
  "alias": "string"
  "lifetime": "integer",
  "attack-status": "string"
 }
```

Figure 11: Efficacy Update

The 'attack-status' field is a mandatory attribute. The various possible values contained in the 'attack-status' field are explained below:

in-progress: DOTS client determines that it is still under attack.

terminated: Attack is successfully mitigated (e.g., attack traffic is dropped).

The DOTS server indicates the result of processing the PUT request using CoAP response codes. The response code 2.04 (Changed) will be returned in the response if the DOTS server has accepted the mitigation efficacy update. If the DOTS server does not find the policy number conveyed in the PUT request in its policy state data then the server MAY accept the mitigation request and will try to mitigate the attack, resulting in a 2.01 (Created) Response Code. The 5.xx response codes are returned if the DOTS server has erred or is incapable of performing the mitigation.

5.3. DOTS Signal Channel Session Configuration

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

a. Heartbeat interval: DOTS agents regularly send heartbeats to each other after mutual authentication in order to keep the DOTS signal channel open.

b. Acceptable signal loss ratio: Maximum retransmissions, retransmission timeout value and other message transmission parameters for the DOTS signal channel.

5.3.1. Discover Acceptable Configuration Parameters

A GET request is used to obtain acceptable configuration parameters on the DOTS server for DOTS signal channel session configuration. Figure 12 shows how to obtain acceptable configuration parameters for the server.

```
Header: GET (Code=0.01)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "config"
```

Figure 12: GET to retrieve configuration

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

```
Content-Format: "application/cbor"
 {
   "heartbeat-interval": {"MinValue": 15, "MaxValue" : 60},
   "max-retransmit": {"MinValue": 3, "MaxValue" : 15},
   "ack-timeout": {"MinValue": 1, "MaxValue" : 30},
   "ack-random-factor": {"MinValue": 1.0, "MaxValue" : 4.0}
  }
```

Figure 13: GET response body

5.3.2. Convey DOTS Signal Channel Session Configuration

A POST request is used to convey the configuration parameters for the signaling channel (e.g., heartbeat interval, maximum retransmissions etc). Message transmission parameters for CoAP are defined in Section 4.8 of [RFC7252]. These parameters can be modified by the DOTS agent (need not be default). If the DOTS client wishes to change the default values of message transmission parameters then it should follow the guidence given in Section 4.8.1 of [RFC7252]. The signaling channel session configuration is applicable to all DOTS signal channel sessions between the DOTS agents.

```
Header: POST (Code=0.02)
Uri-Host: "host"
Uri-Path: ".well-known"
Uri-Path: "version"
Uri-Path: "DOTS-signal"
Uri-Path: "config"
Content-Format: "application/cbor"
{
   "policy-id": "integer",
   "heartbeat-interval": "integer",
   "max-retransmit": "integer",
   "ack-timeout": "integer",
   "ack-random-factor": "number"
}
```

Figure 14: POST to convey the DOTS signal channel session configuration data.

The header fields are described below:

policy-id: An identifier of the policy represented as an integer. This identifier MUST be unique for each policy bound to the DOTS client, i.e., the policy-id needs to be unique relative to the active policies with the DOTS server. This identifier MUST be generated by the DOTS client. This document does not make any assumption about how this identifier is generated. This is a mandatory attribute.

heartbeat-interval: Heartbeat interval to check the DOTS peer health. 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 intial 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.

In the POST request at least one of the attributes heartbeat-interval or max-retransmit or ack-timeout or ack-random- factor MUST be present. The POST request with higher numeric policy identifier value over-rides the DOTS signal channel session configuration data installed by a POST request with a lower numeric policy identifier value.

Figure 15 shows a POST request to convey the configuration parameters for the DOTS signal channel.

```
  Header: POST (Code=0.02)
  Uri-Host: "www.example.com"
  Uri-Path: ".well-known"
  Uri-Path: "v1"
  Uri-Path: "DOTS-signal"
  Uri-Path: "config"
  Content-Format: "application/cbor"
   {
     "policy-id": 1234534333242,
     "heartbeat-interval": 30,
     "max-retransmit": 7,
     "ack-timeout": 5,
     "ack-random-factor": 1.5
    }
```

Figure 15: POST to convey the configuration parameters

The DOTS server indicates the result of processing the POST request using CoAP response codes. The CoAP response will include the CBOR body received in the request. Response code 2.01 (Created) will be returned in the response if the DOTS server has accepted the configuration parameters. If the request is missing one or more mandatory attributes then 4.00 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 4.02 (Invalid query) will be returned in the response. Response code 4.22 (Unprocessable Entity) will be returned in the response if any of the heartbeat-interval, max-retransmit, target-protocol, ack-timeout and ack-random-factor attribute values is not acceptable to the DOTS server. The DOTS server in the error response conveys the minumum and maximum attribute values acceptable by the DOTS server. The DOTS client can re-try and send the POST request with updated attribute values acceptable to the DOTS server.

```
  Content-Format: "application/cbor"
   {
     "policy-id": 1234534333242,
     "heartbeat-interval": {"MinValue": 15, "MaxValue" : 60},
     "max-retransmit": {"MinValue": 3, "MaxValue" : 15},
     "ack-timeout": {"MinValue": 1, "MaxValue" : 30},
     "ack-random-factor": {"MinValue": 1.0, "MaxValue" : 4.0}
    }
```

Figure 16: Error response body

5.3.3. Delete DOTS Signal Channel Session Configuration

A DELETE request is used to delete the installed DOTS signal channel session configuration data (Figure 17).

```
  Header: DELETE (Code=0.04)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "version"
  Uri-Path: "DOTS-signal"
  Uri-Path: "config"
  Content-Format: "application/cbor"
   {
     "policy-id": "integer"
   }
```

Figure 17: DELETE configuration

If the DOTS server does not find the policy number conveyed in the DELETE request in its policy state data, then it responds with a 4.04 (Not Found) error response code. The DOTS server successfully acknowledges a DOTS client’s request to remove the DOTS signal channel session configuration using 2.02 (Deleted) response code.

5.3.4. Retrieving DOTS Signal Channel Session Configuration

A GET request is used to retrieve the installed DOTS signal channel session configuration data from a DOTS server. Figure 18 shows how to retrieve the DOTS signal channel session configuration data.

```
  Header: GET (Code=0.01)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "version"
  Uri-Path: "DOTS-signal"
  Uri-Path: "config"
  Uri-Path: "policy-id value"
```

Figure 18: GET to retrieve configuration

5.4. Redirected Signaling

Redirected Signaling is discussed in detail in Section 3.2.2 of [I-D.ietf-dots-architecture]. If the DOTS server wants to redirect the DOTS client to an alternative DOTS server for a signaling session then the response code 3.00 (alternate server) will be returned in the response to the client.

The DOTS server in the error response conveys the alternate DOTS server FQDN, and the alternate DOTS server IP addresses and TTL (time to live) values in the CBOR body.

```
{
    "alt-server": "string",
    "alt-server-record":
    [
      {
        "addr": "string"
        "TTL" :  "integer",
      }
    ]
}
```

Figure 19: Error response body

The header fields are described below:

alt-server: FQDN of alternate DOTS server.

addr: IP address of alternate DOTS server.

TTL: Time to live represented as an integer number of seconds.

Figure 20 shows a 3.00 response to convey the DOTS alternate server www.example-alt.com, its IP addresses 2002:db8:6401::1 and 2002:db8:6401::2, and TTL values 3600 and 1800.

```
{

    "alt-server": "www.example-alt.com",
    "alt-server-record":
    [
      {
        "TTL" :  3600,
        "addr": "2002:db8:6401::1"
      },
      {
        "TTL" :  1800,
        "addr": "2002:db8:6401::2"
      },

    ]
}
```

Figure 20: Example of error response body

When the DOTS client receives 3.00 response, it considers the current request as having failed, but SHOULD try the request with the alternate DOTS server. During a DDOS attack, the DNS server may be subjected to DDOS attack, alternate DOTS server IP addresses conveyed in the 3.00 response help the DOTS client to skip DNS lookup of the alternate DOTS server and can try to establish UDP or TCP session with the alternate DOTS server IP addresses. The DOTS client SHOULD implement DNS64 function to handle the scenario where IPv6-only DOTS client communicates with IPv4-only alternate DOTS server.

5.5. Heartbeat Mechanism

While the communication between the DOTS agents is quiescent, the DOTS client will probe the DOTS server to ensure it has maintained cryptographic state and vice versa. Such probes can also keep alive firewall or NAT bindings. This probing reduces the frequency of needing a new handshake when a DOTS signal needs to be conveyed to the DOTS server. In DOTS over UDP, heartbeat messages can be exchanged between the DOTS agents using the "COAP ping" mechanism (Section 4.2 in [RFC7252]). The DOTS agent sends an Empty Confirmable message and the peer DOTS agent will respond by sending an Reset message. In DOTS over TCP, heartbeat messages can be exchanged between the DOTS agents using the Ping and Pong messages (Section 4.4 in [I-D.ietf-core-coap-tcp-tls]). The DOTS agent sends an Ping message and the peer DOTS agent will respond by sending an single Pong message.

6. (D)TLS Protocol Profile and Performance considerations

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 machine-in-the-middle and protocol downgrade. These are general attacks on (D)TLS and not specific to DOTS over (D)TLS; please 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 encryption of DOTS using (D)TLS is virtually a green-field deployment DOTS agents MUST implement only (D)TLS 1.2 or later.

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

- DOTS client can use (D)TLS session resumption without server-side state [RFC5077] to resume session and convey the DOTS signal.

- Raw public keys [RFC7250] which reduce the size of the ServerHello, and can be used by servers that cannot obtain certificates (e.g., DOTS gateways on private networks).

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

- TLS False Start [RFC7918] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.

- 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.

- TCP Fast Open [RFC7413] can reduce the number of round-trips to convey DOTS signal.

6.1. MTU and Fragmentation Issues

To avoid DOTS signal message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. If the Path MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP is used to convey the DOTS signal messages then 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 overhead of 13 octets + authentication overhead of the negotiated DTLS cipher suite + block padding (Section 4.1.1.1 of [RFC6347]]. If the 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-ip' field could be split into multiple lists and each list conveyed in a new POST 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 absolutely ensure that there is no IP fragmentation. If IPv4 support on unusual networks is a consideration and path MTU is unknown, implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes, as per [RFC0791] IP packets up to 576 bytes should never need to be fragmented, thus sending a maximum of 500 bytes of DOTS signal over a UDP datagram will generally avoid IP fragmentation.

7. (D)TLS 1.3 considerations

TLS 1.3 [I-D.ietf-tls-tls13] provides critical latency improvements for connection establishment over TLS 1.2. The DTLS 1.3 protocol [I-D.rescorla-tls-dtls13] is based on the TLS 1.3 protocol and provides equivalent security guarantees. (D)TLS 1.3 provides two basic handshake modes of interest to DOTS signal channel:

- Absent packet loss, a full handshake in which the DOTS client is able to send the DOTS signal message after one round trip and the DOTS server immediately after receiving the first DOTS signal message from the client.

- 0-RTT mode in which the DOTS client can authenticate itself and send DOTS signal message on its first flight, 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 SHOULD establish a (D)TLS session with the DOTS server during peacetime and share a PSK. During DDOS attack, the DOTS client can use the (D)TLS session to convey the DOTS signal message and if there is no response from the server after multiple re-tries then the DOTS client can resume the (D)TLS session in 0-RTT mode using PSK. A simplified TLS 1.3 handshake with 0-RTT DOTS signal message exchange is shown in Figure 21.

```
          DOTS Client                                    DOTS Server

         ClientHello
         (Finished)
         (0-RTT DOTS signal message)
         (end_of_early_data)        -------->
                                                        ServerHello
                                               {EncryptedExtensions}
                                               {ServerConfiguration}
                                                       {Certificate}
                                                 {CertificateVerify}
                                                          {Finished}
                                   <--------   [DOTS signal message]
         {Finished}                -------->

         [DOTS signal message]     <------->   [DOTS signal message]
```

Figure 21: TLS 1.3 handshake with 0-RTT

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

(D)TLS based on client certificate can be used for mutual authentication between DOTS agents. If a DOTS gateway is involved, DOTS clients and DOTS gateway MUST perform mutual authentication; only authorized DOTS clients are allowed to send DOTS signals to a DOTS gateway. DOTS gateway and DOTS server MUST perform mutual authentication; DOTS server only allows DOTS signals from authorized DOTS gateway, creating a two-link chain of transitive authentication between the DOTS client and the DOTS server.

```
 +--------------------------------------------------+
 |        example.com domain          +---------+   |
 |                                    | AAA     |   |
 |   +---------------+                | Server  |   |
 |   | Application   |                +------+--+   |
 |   | server        +                       ^
 |   | (DOTS client) |<------------------+   |      |
 |   +---------------+                   +   |      |               example.net
domain
 |                                       V   V      |
 |                               +-------------+    |             +-------------
--+
 |  +--------------+             |             |    |             |
  |
 |  |   Guest      +<-----x----->+             +<---------------->+    DOTS
  |
 |  | (DOTS client)|             |   DOTS      |    |             |    Server
  |
 |  +--------------+             |   Gateway   |    |             |
  |
 |                               +----+--------+    |             +-------------
--+
 |                                    ^             |
 |                                    |             |
 |   +----------------+               |             |
 |   | DDOS detector  |               |             |
 |   | (DOTS client)  +<--------------+             |
 |   +----------------+                             |
 |                                                  |
 +--------------------------------------------------+
```

Figure 22: Example of Authentication and Authorization of DOTS Agents

In the example depicted in Figure 22, the DOTS gateway and DOTS clients within the ’example.com’ domain mutually authenticate with each other. 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 detector to request DDOS mitigation, but does not permit the user of type ’guest’ to request DDOS mitigation.

Also, DOTS gateway and DOTS server MUST perform mutual authentication 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 22, the DOTS server only allows the DOTS gateway to request mitigation for ’example.com’ domain and not for other domains.

9. IANA Considerations

TODO

[TBD: DOTS WG will probably have to do something similar to https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-04, create CBOR DOTS claim registry and register the attributes defined in this specification].

10. Security Considerations

Authenticated encryption MUST be used for data confidentiality and message integrity. (D)TLS based on client certificate MUST be used for mutual authentication. The interaction between the DOTS agents requires Datagram Transport Layer Security (DTLS) and 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.

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]. 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.

Special care should be taken in order to ensure that the activation of the proposed mechanism won’t have an impact on the stability of the network (including connectivity and services delivered over that network).

Involved functional elements in the cooperation system must establish exchange instructions and notification over a secure and authenticated channel. Adequate filters can be enforced to avoid that nodes outside a trusted domain can inject request such as deleting filtering rules. Nevertheless, attacks can be initiated from within the trusted domain if an entity has been corrupted. Adequate means to monitor trusted nodes should also be enabled.

11. Contributors

The following individuals have contributed to this document:

Mike Geller Cisco Systems, Inc. 3250 Florida 33309 USA Email: mgeller@cisco.com

Robert Moskowitz HTT Consulting Oak Park, MI 42837 United States
Email: rgm@htt-consult.com

12. Acknowledgements

Thanks to Christian Jacquenet, Roland Dobbins, Andrew Mortensen, Roman D. Danyliw, and Gilbert Clark for the discussion and comments.

13. References

13.1. Normative References

[I-D.ietf-core-coap-tcp-tls]
Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", draft-ietf-core-coap-tcp-tls-05 (work in progress), October 2016.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7641>.

13.2. Informative References

[I-D.ietf-dots-architecture] Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.ietf-dots-requirements] Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-03 (work in progress), October 2016.

[I-D.ietf-dots-use-cases] Dobbins, R., Fouant, S., Migault, D., Moskowitz, R., Teague, N., and L. Xia, "Use cases for DDoS Open Threat Signaling", draft-ietf-dots-use-cases-02 (work in progress), October 2016.

[I-D.ietf-tls-tls13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-18 (work in progress), October 2016.

[I-D.reddy-dots-data-channel] Reddy, T., Wing, D., Boucadair, M., Nishizuka, K., and L. Xia, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel", draft-reddy-dots-data-channel-01 (work in progress), October 2016.

[I-D.rescorla-tls-dtls13] Rescorla, E. and H. Tschofenig, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-rescorla-tls-dtls13-00 (work in progress), October 2016.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc4732>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <http://www.rfc-editor.org/info/rfc5077>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <http://www.rfc-editor.org/info/rfc6520>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[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, <http://www.rfc-editor.org/info/rfc6724>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <http://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, <http://www.rfc-editor.org/info/rfc7924>.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Dan Wing
USA

Email: dwing-ietf@fuggles.com

DOTS
Internet-Draft
Intended status: Standards Track
Expires: May 21, 2017

N. Teague
Verisign, Inc.
A. Mortensen
Arbor Networks, Inc.

November 17, 2016

DDoS Open Threat Signaling Protocol
draft-teague-dots-protocol-01

Abstract

This document describes Distributed-Denial-of-Service (DDoS) Open Threat Signaling (DOTS), a protocol for requesting and managing mitigation of DDoS attacks.

DOTS mitigation requests over the signal channel permit domains to signal the need for help fending off DDoS attacks, setting the scope and duration of the requested mitigation. Elements called DOTS servers field the signals for help, and enable defensive countermeasures to defend against the attack reported by the clients, reporting the status of the delegated defense to the requesting clients. DOTS clients additionally may use the data channel to manage filters and black- and white-lists to restrict or allow traffic to the clients' domains arbitrarily.

The DOTS signal channel may operate over UDP [RFC0768] and if necessary TCP [RFC0793]. This revision discusses a transport-agnostic approach to this channel, focusing on the message exchanges and delegating transport specifics to other documents. The DOTS data channel operates over HTTPS or a transport with similar reliability, interaction and security characteristics.

Status of This Memo

This Internet-Draft will expire on May 20, 2017.

Copyright Notice

Table of Contents

1. Introduction

Distributed-Denial-of-Service attack scale and frequency continues to increase year over year, and the trend shows no signs of abating [WISR]. In response to the DDoS attack trends, service providers and vendors have developed various approaches to sharing or delegating responsibility for defense, among them ad hoc service relationships, filtering through peering relationships [COMMUNITYFS], and proprietary solutions ([CLOUDSIGNAL], [OPENHYBRID]). Such hybrid approaches to DDoS defense have proven effective, but the heterogeneous methods employed to coordinate DDoS defenses across domain boundaries have necessarily limited their scope and effectiveness, as the mechanisms in one domain have no traction in another.

The DDoS Open Threat Signaling (DOTS) protocol provides a common mechanism to achieve the coordinated attack response previously restricted to custom or proprietary solutions. To meet the needs of network operators facing down modern DDoS attacks, DOTS itself is a hybrid protocol, consisting of a signal channel and a data channel. DOTS uses the signal channel, a lightweight and robust communication layer, to signal the need for mitigation regardless of network conditions, and uses the data channel, an HTTPS [RFC7230] based communication layer with RESTful [REST] semantics, as vehicle for provisioning, configuration, and filter management.

The DOTS protocol is not intended as a replacement for such protocols as BGP Flow Specification [RFC5575] or as a general purpose DDoS countermeasure application programming interface (API), but rather as an advisory protocol enabling attack response coordination between DOTS agents. Any DOTS-enabled device or service is capable of triggering a request for help and shaping the scope and nature of that help, with the details of the actual mitigation left to the discretion of the operators of the attack mitigators. DOTS thereby permits all participating parties to manage their own attack defenses in the manner most appropriate for their own domains.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terms used to define entity relationships, transmitted data, and methods of communication are drawn from the terminology defined in [I-D.ietf-dots-requirements].

2. Architecture

The architecture in which the DOTS protocol operates is assumed to be derived from the architectural components and concepts described in [I-D.ietf-dots-architecture].

2.1. DOTS Agents

All protocol communication is between a DOTS client and a DOTS server. The logical agent termed a DOTS gateway is in practice a DOTS server placed back-to-back with a DOTS client. As discussed in [I-D.ietf-dots-architecture], any interface enabling the back-to-back DOTS server and client to act as a DOTS gateway is implementation-specific. This protocol is therefore concerned only with managing one or more bilateral relationships between DOTS clients and the DOTS servers, a signaling mode known as Direct Signaling in the DOTS architecture. This is shown in Figure 1 below:

```
    +-----------+  signal channel  +-----------+
    |           |<---------------->|           |
    |DOTS client|                  |DOTS server|
    |           |<================>|           |
    +-----------+   data channel   +-----------+
```

Figure 1: DOTS protocol direct signaling

The DOTS architecture anticipates many-to-one and one-to-many deployments, in which multiple DOTS clients maintain distinct signaling sessions with a single DOTS server or a single DOTS client maintains distinct signaling sessions with multiple DOTS servers, as shown below in Figure 2:

```
+----+      +----+      +----+
| c1 |      | Sa |------| c2 |
+----+      +----+      +----+
      \                /
       \              /
        \   +----+   /
         +--| Sb |--+
            +----+

DOTS        DOTS        DOTS
client 1    servers     client 2
```

Figure 2: DOTS protocol direct signaling

DOTS server Sb has signaling sessions with DOTS clients c1 and c2. DOTS client c2 has signaling sessions with DOTS servers Sa and Sb. Except where explicitly defined in this protocol, all mechanisms to maintain multiple signaling sessions are left to the implementation.

3. Protocol Overview

The DOTS protocol consists of two channels, a signal channel and a data channel. The signal channel is the minimal secure communication layer a DOTS client uses to request mitigation for resources under the administrative control of the DOTS client; the administrative control may be delegated. The data channel offers DOTS client operators the limited ability to adjust configuration and filtering for their mitigation requests.

4. Signal Channel

The purpose of the signaling channel is to convey DDoS mitigation request and status information between participating agents (client and server or gateway). Conditions during a DDoS attack are invariably hostile for connection-oriented protocols traversing affected paths. Mechanisms such as Happy Eyeballs [RFC6555] may be used to select a transport suitable for a given time and prevailing network conditions.

Implementations are required to support the DOTS signal channel over UDP as specified in [I-D.ietf-dots-requirements]. This document therefore assumes the the availability of a transport based upon UDP

[RFC0768], but also defines the message exchanges agnostic of the underlying transport used to convey the signaling.

Key tenets of DOTS protocol design are low communication overhead and efficient message packing to increase the chances of successful transmission and receipt. Desirable side-effects of efficient packing are the removal of the possibility of fragmentation in addition to a message size that is friendly towards encapsulation (e.g via GRE [RFC2784] or MPLS [RFC3031]). Large UDP packets may also be treated adversely by middleboxes with restrictive policies or may fall foul of aggressive filtering.

In support of operational requirements for protocol efficiency, backward compatibility and extensibility in [I-D.ietf-dots-requirements], the signaling channel uses Protocol Buffers [PROTOBUF], also known as Protobufs, to define message schemas and serialize messages exchanged between DOTS agents.

Data serialization alone does not cover the security requirements in [I-D.ietf-dots-requirements] of peer mutual authentication (SEC-001), message confidentiality (SEC-002), message replay protection (SEC-003) or message integrity. These qualities must be present in the transport over which the DOTS protocol operates. Key distribution may be achieved via the data channel, via an online mechanism such as DANE [RFC6698], Enrollment over Secure Transport [RFC7030], or by out-of-band means.

4.1. Minimum Viable Information

DOTS is intended to be extensible and to evolve to meet the future needs in communicating as yet unknown threats. However, it must be able to convey the minimum information required for an upstream mitigation platform to successfully counter a DDoS attack. A client may have limited visibility into the full breadth of an attack and as such may not be well placed to provide useful telemetry. DDoS sources may or may not be spoofed and number in the millions. Once mitigation is active, the filtered traffic seen by the DOTS client (or elements informing the DOTS client operator’s decision to request mitigation) may not be representative of the ongoing attack. This provides challenges for the quality and usefulness of telemetry and mitigation/countermeasure stipulations and as such this type of information if conveyed can only be considered advisory.

In these instances the minimum viable information required for the majority of mitigations to be activated is that which pertains to the resource being targeted by the attack (host, prefix, protocol, port, URI etc.), per [I-D.ietf-dots-requirements] (OP-006). The DOTS requirements also identify a mitigation lifetime period (OP-005) and mitigation efficacy metric (OP-007). The former may be considered for inclusion in the minimum viable information set, however, the latter may only be relevant in updates. An explicit mitigation request/terminate flag is also required: a mitigation MUST be explicitly requested by a DOTS client operator. Finally, each message should include a message id or sequence number field as well as a field for the last received message id or sequence number. These may then be compared by the endpoints to assist in tracking state and/or identifying loss.

4.2. Signal Channel Messages

4.2.1. Messaging Overview

The signal channel requirements in [I-D.ietf-dots-requirements] demand a protocol capable of operating despite significant link congestion between DOTS agents. In an effort to maximize the probability of signal delivery between DOTS agents, all DOTS signal channel messages in the DOTS protocol are conceptually unidirectional heartbeats sent between DOTS agents.

The result is a form of scheduled messaging between DOTS agents, in contrast to a conventional request-response model. Once a signal channel is established, a DOTS client begins sending unidirectional heartbeats to the DOTS server, separated by the configured session heartbeat interval (see Section 4.3.1 below). To request mitigation, a DOTS client merely adds the requested mitigation parameters to its heartbeat, and maintains that request until a heartbeat from the DOTS server indicates receipt of that mitigation request. The DOTS server likewise sends its unidirectional heartbeat on the schedule interval, augmenting the content of the heartbeat with mitigation request status.

A simple exchange between DOTS agents with an established signaling session is shown below in Figure 3.

```
   Client                           Server
     |                                 |  // Active signaling session
     |<==== Signal Channel Active ====>|  // with heartbeat interval of
     |                                 |  // 15 seconds, +/- jitter.
     |                                 |
     |----------Heartbeat------------->|  // Client heartbeat
     |                                 |
     |<---------Heartbeat--------------|  // Server heartbeat
     |                                 |
     \                                 \  // 15 seconds elapse, during
     /                                 /  // which client adds request
     \                                 \  // for mitigation.
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat now has
     |          + MitigationRequest    |  // mitigation request params.
     |                                 |
     |<---------HeartBeat--------------|  // Server heartbeat now has
     |          + MitigationPending    |  // pending mitigation request
     |                                 |  // status.
     |                                 |
     \                                 \
     /                                 /  // 15s elapse, +/- jitter.
     \                                 \
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat maintains
     |          + MitigationRequest    |  // request as it waits for
     |                                 |  // mitigation to begin.
     |                                 |
     |     X----HeartBeat--------------|  // Mitigation enabled, but
     |          + MitigationRunning    |  // server heartbeat is dropped
     |                                 |
     \                                 \
     /                                 /  // 15s elapse, +/- jitter.
     \                                 \
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat maintains
     |          + MitigationRequest    |  // request as it waits for
     |                                 |  // mitigation to begin.
     |                                 |
     |<---------HeartBeat--------------|  // Server heartbeat, status of
     |          + MitigationRunning    |  // running mitigation
     |                                 |  // delivered to client.
```

Figure 3: Signaling Session with Mitigation Request

All messages are variations of this scheduled heartbeat model. See Section 4.3 below for detailed discussion of the message exchanges between DOTS agents.

4.2.2. Message Definition and Serialization

The DOTS protocol signal channel uses Protobufs [PROTOBUF] as an interface definition language (IDL) for signal channel messaging between DOTS agent, reducing the number of discrete messages to just a single message superset per direction, with function defined by the chosen fields contained within the message.

Protobufs schemas are used to define the messages sent in either direction, from which code may be generated for specific language implementations of DOTS. As Protobufs serialization relies on numbered fields, signal channel messaging permits the introduction of new numbered fields arbitrarily, adding the requisite extensibility to the protocol while retaining backward compatibility. Future revisions of or extensions to the protocol may use the data channel to provide a mechanism by which schema updates or expansions may be communicated during provisioning/session establishment.

4.2.3. Client Message Fields

Only the seqno and last_svr_seqno fields are required in every message, as they are the minimum required for the heartbeat. Subsets of the other fields are used to convey a given signal message type.

The fields in the DOTS client signal channel message have the following functions:

seqno: a client-generated sequence number unique to the message. The client increments the seqno value by one for each message sent over the signal channel.

last_svr_seqno: the sequence number of the last message received from the server, provided to the server as a simple way to detect lost messages.

mitigations: a list of mitigations requested or withdrawn by the client. The mitigation schema fields are described below.

active: indicates a request for a list of active mitigations and their detail that are current on the DOTS server.

ping: an operator initiated heartbeat like message which will ellicit a response from the DOTS server. This may be used to prove bi-directional communications on an ad-hoc basis. For example, a DOTS ping may be used to prove keying material on the DOTS client is valid and may be used to establish signaling sessions with the DOTS server.

extensions: these fields may be used to communicate implementation specific details. An example would be the dissemination of filters between DOTS client and DOTS server.

4.2.3.1. Client Message Schema

The DOTS client message schema is detailed in Section 8.1.

4.2.4. Mitigation Request Fields

The fields in a mitigation request are as follows:

eventid: an opaque client generated identifier that distinguishes a unique event or incident. May be used by the client as a reference to the specific event triggering a mitigation request, or for other implementation-specific purposes.

requested: signals the need for mitigation to the DOTS server. If true, the DOTS client is requesting mitigation for the provided scope. If false, the DOTS client is indicating it does not require mitigation, and the DOTS server MUST cease the mitigation for the provided scope.

scope: the scope of the mitigation requested, which may be any of the types described in [I-D.ietf-dots-requirements], such as Classless Internet Domain Routing (CIDR) [RFC1518],[RFC1519] prefixes, DNS names, or aliases defined by the DOTS client operator through the data channel.

lifetime: the lifetime in seconds a mitigation request should be considered valid.

efficacy: a metric to convey to a DOTS server the perceived efficacy of an active mitigation, per operational requirements in [I-D.ietf-dots-requirements]. The mitigation efficacy is represented as a floating point value between 0 and 1, with smaller values indicating lesser efficacy, and larger greater efficacy.

extensions: these fields may be used to provide implementation-specific mitigation details.

4.2.4.1. Mitigation Request Schema

The DOTS client message schema is detailed in Section 8.2.

4.2.5. DOTS Server Message Fields

DOTS server messages use a subset of the available fields to convey the given signal type, including additional relevant fields as necessary. The only fields which may be common to all signals are seqno and last_client_seqno which may be used to detect message loss or out-of-order delivery. When conveying mitigation information, the server schema may bundle multiple mitigation status datasets into a single message, provided this does not violate the required sub-MTU message size [I-D.ietf-dots-requirements].

The fields in the DOTS server signal channel message schema have the following functions:

seqno: a server generated sequence number unique to the message.

last_cli_seqno: the seqno of the last message received from the client.

ping: an operator-initiated heartbeat like message which will ellicit a response from the DOTS client. This may be used to prove bi-directional communications on an ad-hoc basis.

error: details of an error caused by a DOTS client request.

redirect: Populated with the details of the redirection target DOTS server, if the DOTS server is redirecting the DOTS client to another DOTS server.

mitigations: a list containing the status of mitigations requested by the DOTS client. The fields in the mitigation status schema are described below.

extensions: these fields may be used to communicate implementation specific details. An example would be the communication of DNS mitigation vip to the DOTS client by the DOTS server.

4.2.5.1. DOTS Server Message Schema

The server message schema is detailed in Section 8.4.

4.2.6. Server Errors

If a DOTS client message cannot be processed by the DOTS server, or for any other reason causes an error, the DOTS server MUST populate the error field in any response to the message causing the error. As the error response itself may be lost, a DOTS client may continue sending problematic messages regardless of the DOTS server’s error notifications. DOTS server implementations MAY terminate the signaling session after client-triggered errors exceed a threshold during a time period equivalent to three times the session heartbeat interval.

The DOTS client message triggering the error condition is indicated in the last_client_seqno value of the DOTS server message containing the error.

Error codes MUST be one of the following types:

NOERROR: Indicates the DOTS server has detected no error resulting from a DOTS client message. Implementations MAY omit the error field entirely when no error condition is present. This value is included in the schema largely to adhere to the convention that an error status of 0 indicates success.

INVALID_VALUE: Indicates the DOTS client included an invalid value for a field in the client message most recently received from the client. The DOTS server SHOULD include specifics of the invalid value in the details field of the error.

MITIGATION_UNAVAILABLE: Indicates the DOTS server is unable to provide mitigation in response to a mitigation request from the DOTS client.

MITIGATION_CONFLICT: Indicates a mitigation request conflicts with an existing mitigation from the client. The DOTS server SHOULD populate the error details field with the status information of the mitigation conflicting with the requested mitigation.

MALFORMED_MESSAGE: Indicates the DOTS client message is malformed and cannot be processed.

4.2.6.1. Server Error Fields

code: a numeric code categorizing the error type detected by the DOTS server.

details: specific information about the reason for the detected error.

4.2.6.2. Server Error Schema

The server error schema is detailed in Section 8.7.

4.2.7. Server Mitigation Status Fields

The DOTS server message contains zero or more mitigation status messages, the fields of which have the following functions:

eventid: an opaque client generated identifier that distinguishes a unique event or incident.

ttl: the remaining lifetime of the mitigation, in seconds.

bytes_dropped: the total dropped byte count for the mitigation associated with eventid.

bps_dropped: the dropped bytes per second for the mitigation associated with eventid. This value is expected to be calculated by the mitigator, and as such is implementation-specific.

pkts_dropped: the total dropped packet count for the mitigation associated with eventid..

pps_dropped: the dropped packets per second for the mitigation associated with eventid. This value is expected to be calculated by the mitigator, and as such is implementation-specific.

blacklist_enabled: Indicates whether a blacklist of prohibited traffic sources is enabled for the mitigation associated with eventid. The blacklist is managed through the data channel.

whitelist_enabled: Indicates whether a whitelist of sources from which traffic must always be allowed is enabled. The whitelist is managed through the data channel.

filters_enabled: Indicates whether client-specified traffic filters are enabled for the mitigation associated with eventid.

4.3. Interactions

4.3.1. Session Initialization

Signaling sessions are initiated by the DOTS client. Session initialization begins when the DOTS client connects to the DOTS server port, 4646. After connecting, the DOTS client establishes the channel security context, including all necessary cryptographic exchanges between the two DOTS agents.

This signal channel specification is transport-agnostic, and delegates the details of transport, including transport security, to transport-specific documents. Regardless of transport, DOTS implementations nonetheless MUST provide signal channel security meeting the requirements in [I-D.ietf-dots-requirements].

Once the signal channel security context is established, the DOTS client sends a channel initialization message to the DOTS server, optionally including signaling session configuration values; if the session configuration values are excluded, defaults MUST be used for the signaling session. An example initialization message setting the acceptable signal loss and heartbeat interval for the signaling sessions is described in Figure 4 below:

```
    message DOTSClientMessage {
      1 (seqno) = %;
      2 (last_svr_seqno) = %;
      6 (config) = {
        1 (loss_limit) = %;
        3 (heartbeat_interval) = %;
      };
    }
```

Figure 4: Signal Channel Initialization Message

The DOTS server MUST respond immediately by sending a heartbeat (see Section 4.3.2 below) to the DOTS client. The signal channel is active when the DOTS client receives a heartbeat from the DOTS server with a last_client_seqno of a signal channel initialization message. Both DOTS agents MUST begin sending heartbeats on the interval for the signaling session once the session is active.

The following example assumes a DOTS implementation using UDP as the transport and DTLS1.2 [RFC6347]. In Figure 5 below, the DOTS client uses the default values for acceptable signal loss, maximum mitigation lifetime, and heartbeat interval. The initial DOTS server heartbeat is lost, so the DOTS client sends another channel initialization message after waiting for the minimum heartbeat interval defined below in Section 4.3.2:

```
   Client                             Server
     |                                 |
     |<- - - - DTLS1.2 handshake - - ->|  // Server listens on UDP:4646.
     |                                 |  // Client initiates DTLS
     |                                 |  // handshake.
     |                                 |
     |----------ChannelInit----------->|  // Client sends signal
     |          seqno = 1              |  // channel init message
     |          last_svr_seqno = 0     |
     |                                 |
     |     X----HeartBeat--------------|  // Server immediately sends
     |          seqno = 1              |  // heartbeat reply, which
     |          last_client_seqno = 1  |  // is lost.
     |                                 |
     \                                 \  // Session heartbeat interval
     /                                 /  // elapses.
     \                                 \
     |----------ChannelInit----------->|  // Client retries signal
     |          seqno = 2              |  // channel init message.
     |          last_svr_seqno = 0     |
     |                                 |
     |<---------HeartBeat--------------|  // Server immediately sends
     |          seqno = 2              |  // heartbeat reply
     |          last_client_seqno = 2  |
     |                                 |
     |<==== Signal Channel Active ====>|
```

Figure 5: Signal Channel Initialization

4.3.1.1. Session Initialization Error Handling

If the DOTS client specifies invalid values for the signal channel configuration, the DOTS server replies with an error, and may ultimately terminate the connection if the client fails to correct the invalid values, as described in [I-D.ietf-dots-architecture].

4.3.1.2. Mis-Sequencing

In the event that the DOTS agent receives messages containing invalid seqno, last_client_seqno or last_svr_seqno these should be discarded and ignored.

4.3.2. Heartbeat

The most common message exchanged between a DOTS client and a DOTS server is a heartbeat (OP-002 [I-D.ietf-dots-requirements]), which maintains and monitors the health of the DOTS session. This is achieved with simple, loosely-coupled bi-directional messages containing the sending DOTS agent’s message sequence number and the sequence number the sending DOTS agent last received from its peer. Due to the stress volumetric DDoS impose upon a network, a degree of loss during attacks is to be expected. Message loss tolerance may be set on signal channel establishment.

The default heartbeat interval is 20 seconds, plus or minus a number of milliseconds between 50 and 2000. The number of milliseconds MUST be randomized in order to introduce jitter into the heartbeat interval, as recommended by [RFC5405]. The default interval is derived from the recommendations in [RFC5405] regarding middlebox traversal, to maintain NAT bindings in the path between DOTS agents.

The interval between heartbeats is may also be set by the client when establishing the signal channel. The minimum heartbeat interval is 15 seconds, plus the random number of milliseconds as described above. The maximum heartbeat interval is 120 seconds (two minutes), minus the random number of milliseconds described above.

Heartbeats are loosely-coupled, meaning each DOTS agent in a bilateral signaling session sends DOTS heartbeats on the specified interval, but asynchronously, without acknowledgement. Each DOTS agent tracks heartbeats received from its peer, and includes the sequence number of the last heartbeat received from the peer agent in the next heartbeat sent, as shown in Figure 6:

```
     Client                           Server
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 1              |
       |          last_svr_seqno = 0     |
       |                                 |
       |<---------HeartBeat--------------|  // Server heartbeat
       |          seqno = 1              |
       |          last_client_seqno = 1  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 2              |
       |          last_svr_seqno = 1     |
       |                                 |
       |      X---HeartBeat--------------|  // Server heartbeat lost
       |          seqno = 2              |
       |          last_client_seqno = 2  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat,
       |          seqno = 3              |  // last_svr_seqno remains 1,
       |          last_svr_seqno = 1     |  // indicating lost heartbeat
       |                                 |
       |<---------HeartBeat--------------|  // Server heartbeat resumes
       |          seqno = 3              |
       |          last_client_seqno = 3  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 4              |
       |          last_svr_seqno = 3     |
       |                                 |
```

Figure 6: Heartbeats Between DOTS agents

The DOTS client heartbeat has the following format:

```
    message DOTSClientMessage {
      1 (seqno) = %;
      2 (last_svr_seqno) = %;
    }
```

The DOTS server heartbeat is identical aside from the schema type:

```
    message DOTSServerMessage {
      1 (seqno) = %;
      2 (last_svr_seqno) = %;
    }
```

Should the number of signals lost exceed the acceptable lossiness value for the signaling session, the agent detecting the signal loss may consider the signaling session lost. The default value for acceptable signal loss is 9, which, when coupled with the default heartbeat interval, amounts to lack of heartbeat from the peer DOTS agent for 180 seconds (three minutes).

4.3.2.1. Ping

There may be cases where a DOTS client or server operator wishes to trigger an immediate heartbeat response in order to validate bi-directional communication (e.g. during provisioning). This ad-hoc triggering may be achieved by setting the ping field set to TRUE. When DOTS agent receives a message on the signal channel with the ping field set to TRUE, it MUST immediately send heartbeat back to the ping sender. A ping reply MUST consist of only the senders sequence number and the sequence number of the received ping. [[EDITOR’S NOTE: rate limiting of pings required?]]

A ping is identical to a standard heartbeat, but with the the ping field included and set to true:

```
message DOTSClientMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
  5 (ping) = true;
}
```

4.3.3. Mitigation Request Handling

The mitigation request is the crux of the DOTS protocol, and is comprised of the minimum viable information described in Section 4.1. The request may be augmented with additional implementation specific extensions where these do not result in undue packet bloat. The DOTS client may send repeated requests until it receives a suitable response from the DOTS server by which it may interpret successful receipt.

```
       message DOTSClientMessage {
         1 (seqno) = %;
         2 (last_svr_seqno) = %;
         3 (mitigations) = [
           {
             1 (eventid) = %;
             2 (requested) = %;
             3 (scope) = %;
             4 (lifetime) = %;
           }
         ];
       }
```

The DOTS server is expected to respond to confirm that it has accepted and or rejected the mitigation request. Upon receipt of the response the DOTS client should cease sending additional initial requests for the same eventid. If these do not cease then the server may assume that the response was possibly lost and should resend accordingly. Acceptance status is communicated by the DOTS server replying with the corresponding eventid and the enabled field set to 1 for acceptance and 0 for rejection. A rejection by the DOTS server should be accompanied with an extension field detailing succinctly the reason (e.g. out of contract, conflict, maintenance etc.).

```
      message DOTSServerMessage {
        1 (seqno) = %;
        2 (last_cli_seqno) = %;
        4 (mitigations) = [
          {
            1 (eventid) = %;
            2 (enabled) = true; // Mitigation request accepted
          }
        ]
      }
```

After a period of time the mitigation request may expire and the DOTS server may end the mitigation. Alternately, the DOTS client may explicitly terminate the active mitigation by sending a message to the server that contains a mitigation value with the eventid and that has the requested field set to false, as shown below:

```
      message DOTSClientMessage {
        1 (seqno) = %;
        2 (last_svr_seqno) = %;
        3 (mitigations) = [
          {
            1 (eventid) = %;
            2 (requested) = false; // Terminate mitigation
          }
        ];
      }
```

The server must explicitly acknowledge the termination with a response message with the enabled field now set to false:

```
      message DOTSServerMessage {
        1 (seqno) = %;
        2 (last_cli_seqno) = %;
        6 (mitigations) = [
          {
            1 (eventid) = %;
            2 (enabled) = false; // Mitigation terminated
          }
        ];
      }
```

The life cycle of a DOTS mitigation request resembles the following:

```
       Client                        Server
         |                              |
         |---Request(M=true)----------->|  // Mitigation request
         |                              |
         |<---------MitigationActive----|  // Server acceptance
         |                              |
         |< - - - - MitigationFeedback -|
         |                              |
         |---Terminate(M=false)-------->|  // Mitigation termination
         |                              |
         |<---------MitigationEnded-----|  // Server termination ack
```

4.3.4. Ancillary Messages

In addition to the basic interaction, additional messages may be exchanged throughout the lifetime of the mitigation. The following message types are defined to provide requisite information between DOTS agents during an active signaling session.

4.3.4.1. Mitigation Feedback

The DOTS server MUST update the client with current mitigation status. This MUST include the eventid, and SHOULD include available dropped attack traffic statistics provided by the mitigator. A DOTS server MAY provide feedback for more than one mitigation in a single message, provided the resulting message meets the sub-MTU size requirements in [I-D.ietf-dots-requirements].

The DOTS client SHOULD use the feedback from the DOTS server when deciding to update or terminate a mitigation request. For example, if the DOTS client learns from DOTS server mitigation feedback that the dropped_pps rate is low, the DOTS client might decide to terminate upstream mitigation and handle the attack locally.

A mitigation feedback message from the DOTS server would resemble the following format, assuming an active mitigation request from the DOTS client:

```
message DOTSServerMessage {
  1 (seqno) = %;
  2 (last_client_seqno) = %;
  6 (mitigations) = [
    {
      1 (eventid) = %;
      2 (enabled) = %;
      3 (ttl) = %;
      4 (bytes_dropped) = %;
      5 (bps_dropped) = %;
      6 (pkts_dropped) = %;
      7 (pps_dropped) = %;
      10 (filters_enabled) = true;
    },
  ];
}
```

4.3.4.2. Mitigation Lifetime Update

The DOTS client may wish to update the mitigation during its lifetime. Updates may be to alter the lifetime to extend the mitigation, or an update may communicate the perceived efficacy of the mitigation. The former may be as a result of the DOTS sever feedback which may suggest that an attack shows no sign of abating. The latter may be to notify the DOTS server whether the countermeasures deployed are perceived as effective or not.

A DOTS client may update the lifetime of multiple mitigations in a single request as long as the message size meets the sub-MTU requirement per [I-D.ietf-dots-requirements]. The lifetime update message has the following format:

```
message DOTSClientMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
  3 (mitigations) = [
    {
      1 (eventid) = %;
      2 (requested) = true;
      4 (lifetime) = %;
    }
  ];
}
```

Upon receipt of the mitigation lifetime update, the DOTS server replace the current mitigation expiration time with the new value. The updated lifetime MUST be visible in the ttl field in subsequent mitigation feedback messages. When updating a mitigation lifetime, the DOTS client SHOULD continue sending the lifetime update request at the heartbeat interval until the DOTS server’s mitigation feedback shows an updated ttl for the updated mitigation.

4.3.4.3. Mitigation Efficacy Updates

When a mitigation is active, a DOTS client MUST periodically communicate the locally perceived efficacy of the mitigation to the DOTS server. This gives the DOTS server a rough sense of whether the DOTS client perceives the mitigator’s deployed countermeasures as effective. The efficacy update update message has the following format:

```
message DOTSClientMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
  3 (mitigations) = [
    {
      1 (eventid) = %;
      6 (efficacy) = %;
    }
  ];
}
```

The DOTS server SHOULD consider the efficacy update an indication of the effectiveness of any ongoing mitigations related to the eventid provided by the DOTS client. The DOTS server nonetheless MAY treat any efficacy update from the client as advisory, and is under no obligation to alter the mitigation strategy in response.

5. Data Channel

5.1. Role

Using the conventions established in [REST], the data channel provides an interface for configuration, black- and white-list management, traffic filter management, and extensibility required for future operator needs (GEN-001 [I-D.ietf-dots-requirements]).

5.2. Limitations

Unlike the DOTS signal channel, the data channel potentially offers DOTS client operators limited direct control over the behavior of mitigations requested by the DOTS client. However, the DOTS data channel is not a general purpose application programming interface for mitigators with which a DOTS server is communicating. Certain countermeasure profiles for DDoS attacks are widely understood and deployed, but many remain specific to mitigation vendor implementations, making abstraction all but impossible. The DOTS data channel in this protocol is therefore focused on a limited subset of widely available and well understood mitigation actions, namely black- and white-listing, and rate-limiting.

While managing filters and rate-limit policy over the DOTS data channel resembles the dissemination of flow specifications with a match and action on match in [RFC5575], the similarity is restricted to [RFC5575]’s traffic-rate action only in order to prevent a DOTS client from exerting influence over traffic not destined for the DOTS client’s domain.

5.3. Transport

The DOTS data channel relies on the semantics described in [REST], meaning any reliable application protocol enabling those semantics could be used. This document anticipates HTTP/1.1 over TLS [RFC7230] will be most widely deployed at the time of writing. Implementations of the DOTS protocol therefore MUST support data channels using HTTP/1.1 over TLS. However, this document also leaves open the possibility that the data channel MAY be implemented through such application transports as HTTP/2 [RFC7540] or the Quick UDP Internet Connection [I-D.hamilton-quic-transport-protocol] protocol, as well as other current and future protocols supporting [REST] semantics and the security requirements described in [I-D.ietf-dots-requirements]. Support for alternative secure REST transports for the data channel are deployment- and implementation-specific.

DOTS data channel implementations MUST support the IPv4 [RFC0791] and IPv6 [RFC2460] protocols, and MUST support the "Happy Eyeballs" algorithm for dual stack deployments discussed in [RFC6555].

Implementations of the DOTS data channel MUST use TLS version 1.2 or higher. DOTS agents MUST NOT create a data channel with a peer agent requesting a lower TLS version, and SHOULD drop the connection immediately on detecting the peer DOTS agent does not support a required TLS version.

Section 7 offers a more detailed discussion of data channel transport security, including cipher suites.

5.4. Authentication

When establishing the data channel, the DOTS client and DOTS server MUST mutually authenticate each other, per SEC-001 in [I-D.ietf-dots-requirements]. A common method for mutual authentication for HTTP/1.1 over TLS is an exchange of X.509 certificates between client and server during the TLS handshake [RFC5246]; similar mechanisms exist in HTTP/2 [RFC7540] and in [I-D.hamilton-quic-transport-protocol].

Regardless of the underlying transport used, this document does not prescribe the method of mutual authentication, and alternatives may include a mix of things like basic auth [RFC7617] and HTTP SPNEGO [RFC4559]. The method of mutual authentication used for the data channel is left to the discretion of the DOTS server operator. Additional discussion of mutual authentication is below in Section 7.

5.5. Authorization

TBD deployment-specific, see also security considerations.

5.6. Resources

The DOTS server exposes data channel resources to the DOTS client as uniform resource identifiers. The DOTS client sends requests related to the data channel resources using the verbs defined in [RFC7231]: GET, POST, PUT, PATCH and DELETE. The DOTS server responds to the DOTS client requests with a status code and, if the request succeeded, available data returned by the request. The status codes used in DOTS server responses are also defined in [RFC7231].

5.6.1. Resource Root

The root resource or endpoint in the DOTS data channel is /dots/v1/data. The root resource MUST be prefixed to all resources exposed through the data channel.

5.6.2. {+dataroot}/sessions

The /sessions endpoint is a read-only resource from which the DOTS client may request the status of signaling sessions.

5.6.2.1. GET {+dataroot}/sessions

The DOTS client requests the list of signaling sessions by issuing a GET for the /sessions resource:

```
GET /dots/v1/data/sessions HTTP/1.1
Host: dots-server.example.com
Accept: application/json
```

Figure 7: DOTS Client Requesting Session Status

If the DOTS client is authorized, the DOTS server responds to the GET with a list of signaling session identifiers, as in the following example:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/json

{
    "sessions": [
        {
            "id": <string>,
            "client": <ip_address>,
            "server": <ip_address>,
            "duration": <iso8601_duration>,
        },
        {
            ...
        }
    ]
}
```

The top-level JSON key-value pairs in the response are as follows:

sessions: A list of dictionary objects describing active signaling sessions. If empty, no signaling sessions are active.

Each dictionary within the sessions list contains the following JSON key-value pairs:

id: An opaque alphanumeric string identifying the signaling session.

client: The dotted-quad IPv4 or formatted IPv6 address [RFC5952] of the DOTS client in the signaling session.

server: The dotted-quad IPv4 or formatted IPv6 address [RFC5952] of the DOTS server in the signaling session.

duration: An ISO 8601 representation of the duration of the signaling session.

5.6.3. {+dataroot}/filters

The /filters endpoint on a DOTS server is a read-write resource through which a DOTS client may request that the DOTS server add, retrieve, modify and delete traffic filters to an active mitigation requested through the signal channel.

DOTS servers SHOULD indicate lack of support for filtering by returning a 501 Not Implemented status to any request for a filters URI. If a DOTS client attempts to apply a filter to flows which the DOTS server determines do not belong to the DOTS client, the DOTS server MUST respond with a 403 Forbidden.

A filter is a match and an action on match. As discussed above in Section 5.2, actions are restricted to black- and white-listing and rate-limiting. Matches in a filter dictionary may be any of the match types discussed below. All matches MUST include a destination address or identifier; DOTS server implementations MUST NOT accept filters missing a destination address or prefix.

A filter can be represented as a map or dictionary with the following attributes:

id: a client-generated integer value acting as a unique identifier for the filter.

af: address family of the flow to filter, must be one of "ipv4" or "ipv6". This attribute is required in all filters.

src: source prefix of the flow(s) to filter.

sport: source port of the flow(s) to filter.

dst: destination prefix of the flow(s) to filter.

dport: destination port of the flow(s) to filter.

action: The action to apply to flows matching the filter. The action MUST be one of "blacklist" (i.e., drop all matching flows), "whitelist" (i.e., always forward traffic matching the filter), or "rate-limit" (i.e., control the rate of traffic matching the filter).

Bps: an integer value setting the bytes per second limit for flows matching the filter when action is "rate-limit".

5.6.3.1. POST {+dataroot}/filters/{+mitigation-id}

A POST request over the data channel to the /filters endpoint on a DOTS server permits a DOTS client to manage filtering policy for a mitigation:

```
POST /dots/v1/data/filters/42 HTTP/1.1
Host: dots-server.example.com
Accept: application/json
Content-Type: application/json
Content-Length: NNNN

{
    "filters": [
        {
            "id": 1,
            "af": "ipv4",
            "src": "192.0.2.2/32",
            "action": "blacklist",
        },

        {
            "id": 2,
            "af": "ipv4",
            "src": "192.51.100.0/30",
            "sport": 53,
            "action": "whitelist",
        },

        ...
    ]
}
```

Figure 8: Filter creation

The DOTS server confirms filter creation with an empty OK:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 0
```

5.6.3.2. PUT {+dataroot}/filters/{+mitigation-id}/{+filter-id}

Filters may be updated by sending a PUT request to the specific filter URI. DOTS servers MUST replace the existing filter atomically with the values in the PUT.

```
PUT /dots/v1/data/filters/42/1 HTTP/1.1
Host: dots-server.example.com
Accept: application/json
Content-Type: application/json
Content-Length: NNNN

{
    "id": 1,
    "af": "ipv4",
    "src": "192.0.2.2/32",
    "dst": "198.51.100.0/24",
    "action": "blacklist",
}
```

Figure 9: Filter Update

The DOTS server confirms filter update with a No Content response:

```
HTTP/1.1 204 No Content
Cache-Control: no-cache
Content-Length: 0
```

5.6.3.3. GET {+dataroot}/filters/{+mitigation-id}

A GET request to the /filters endpoint on a DOTS server returns filters for a mitigation requested by the DOTS client. The mitigation-id value MUST be the DOTS client-generated mitigation ID used in a mitigation request previously sent to the DOTS server over the signal channel, with the exception of the global filter list as described below. A request listing the filters active during a mitigation is shown below in Figure 10:

```
GET /dots/v1/data/filters/42 HTTP/1.1
Host: dots-server.example.com
Accept: application/json
```

Figure 10: Filter GET

The DOTS server returns a list of active filters applied as part of the mitigation on the DOTS client’s behalf as in Figure 11:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/json

{
    "id": 42,
    "filters": [
        {
            "id": 1,
            "af": "ipv4",
            "src": "192.0.2.2/32",
            "action": "blacklist",
        },

        {
            "id": 2,
            "af": "ipv4",
            "src": "192.51.100.0/30",
            "sport": 53,
            "action": "whitelist",
        },
    ]
}
```

Figure 11: Filter GET Response

If the filter list is empty, no filters are applied as part of the mitigation.

5.6.4. {+dataroot}/config

The /config data channel endpoint on a DOTS server is a read-write resource through which a DOTS client may configure global signaling session behavior.

5.6.4.1. GET {+dataroot}/config

A GET request to the /config endpoint returns the current DOTS configuration for the DOTS client:

```
GET /dots/v1/data/config HTTP/1.1
Host: dots-server.example.com
Accept: application/json
```

Figure 12: DOTS Client Requesting Configuration

```
HTTP/1.1 200 OK
Cache-Control:
Content-Type: application/json

{
    "config": {
        "protected-resources": {
            <alnum_id>: [
            ]
        }
    }
}
```

5.6.4.2. POST {+dataroot}/config/protected-resources/

TBD

5.6.5. Serialization

Resource data is exchanged between DOTS client in a serialized format. Implementations MUST support JSON [RFC7159] serialization of resource data. DOTS clients MUST advertise support for JSON-encoded data from the DOTS server through the HTTP Accept header [RFC7231] (or an equivalent if not using HTTP), using the MIME type defined in [RFC7159], application/json:

```
        GET /dots/v1/data/sessions HTTP/1.1
        Host: dots-server.example.com
        Accept: application/json
```

Figure 13: DOTS Client Advertising Required Serialization

Implementations MAY offer additional serialization formats as well. DOTS clients MAY advertise support for additional serialization formats in requests to the DOTS server through the HTTP Accept header

[RFC7231] (or an equivalent if not using HTTP), as shown in the example HTTP/1.1 request below:

```
GET /dots/v1/data/sessions HTTP/1.1
Host: dots-server.example.com
Accept: application/json; q=0.5, application/cbor
```

Figure 14: DOTS Client Supporting Additional Serializations

If a DOTS server does not support the media types in the DOTS client’s Accept header (or its equivalent), the DOTS server MUST respond with an status code indicating an error in the client request. In HTTP deployments, the DOTS server MUST return the 415 Unsupported Media Type error code defined in [RFC7231]. A DOTS client request lacking indicated support for application/json content suggests an invalid or malicious client implementation. After sending the 415 error response, DOTS servers SHOULD terminate the data channel connection with the invalid client.

5.6.6. Caching

DOTS server responses sent over the DOTS data channel MUST NOT be cached by the DOTS client. DOTS server implementations therefore MUST include in responses a Cache-Control header with a value of "no-cache" [RFC7234].

6. IANA Considerations

The DOTS protocol requires a well-known port to which DOTS client messages are sent. The DOTS server listens on the well-known port for client messages. The DOTS client binds to an ephemeral port per Section 4.3.1 above. This document selects port 4646 (the ASCII decimal values for "..": "DOTS") as the well-known port.

7. Security Considerations

7.1. Data Channel Security

The DOTS data channel acts as a management plane for DOTS signaling sessions. As discussed in the security considerations of [I-D.ietf-dots-architecture], an attacker with control over data channel may be able to blacklist or rate-limit any flows under the administrative control of the DOTS client. Extra care must therefore be taken when authenticating and authorizing the data channel.

DOTS server operators SHOULD enforce access control policies restricting which clients are able to contact DOTS servers.

7.2. Signal Channel Security

The DOTS signal channel controls mitigation request and withdrawal and as such care must be taken to protect against concerns outlined in the security considerations of [I-D.ietf-dots-architecture].

8. Appendix A: Message Schemas

8.1. DOTS Client Message Schema

```
syntax = "proto3";
import "google/protobuf/any.proto";

message DOTSClientMessage {
  // Client generated sequence number
  uint64 seqno = 1;

  // Sequence number of last received server message
  uint64 last_svr_seqno = 2;

  repeated DOTSMitigation mitigations = 3;

  // Request active mitigation list from server
  bool active = 4;

  // Ping request (operator initiated)
  bool ping = 5;

  DOTSSessionConfig config = 6;

  repeated google.protobuf.Any extensions = 999;
}
```

Figure 15: DOTS Client Message Schema

8.2. Mitigation Request Schema

```
       message DOTSMitigation {
         // Opaque client-generated event identifier
         string eventid = 1;

         // Toggle mitigation for the above scope
         bool requested = 2;

         // Mitigation scope as described in I-D.ietf-dots-requirements
         string scope = 3;

         // Lifetime of the requested mitigation.
         uint32 lifetime = 4;

         // Mitigation efficacy score as a float value between 0 and 1
         float efficacy = 5;

         repeated google.protobuf.Any extensions = 999;
       }
```

Figure 16: DOTS Client Mitigation Request Schema

8.3. Session Configuration Schema

```
         // Per session configuration sent on signaling session init
         message DOTSSessionConfig {
           // Acceptable signal loss
           uint32 loss_limit = 1;

           // Maximum mitigation lifetime in seconds
           uint32 lifetime_max = 2;

           // Heartbeat interval in milliseconds
           uint32 heartbeat_interval = 3;
         }
```

Figure 17: DOTS Session Conifiguration Schema

8.4. DOTS Server Message Schema

```
syntax = "proto3";
import "google/protobuf/any.proto";

message DOTSServerMessage {
  // Server generated sequence number
  uint64 seqno = 1;

  // Sequence number of last received Client message
  uint64 last_client_seqno = 2;

  // Request immediate heartbeat response from client.
  bool ping = 3;

  // Server error details, if available
  DOTSServerError error = 4;

  DOTSRedirect redirect = 5;

  // Mitigation data, limited by MTU
  repeated DOTSMitigationStatus mitigations = 6;
}
```

Figure 18: DOTS Server Message Schema

8.5. DOTS Redirect Schema

```
message DOTSRedirect {
  // Redirection target DOTS server address
  string target = 1;

  // Address family of redirection target
  enum RedirectionTargetType {
    DNSNAME = 0;
    IPV4 = 4;
    IPV6 = 6;
  }
  RedirectionTargetType target_type = 2;

  // Port on which to contact redirection target.
  // XXX Protobufs has no uint16 type, implementations
  // will need to sanity check.
  uint32 port = 3;
}
```

8.6. DOTS Mitigation Status Schema

```
     syntax = "proto3";
     import "google/protobuf/any.proto";

     message DOTSMitigationStatus {
       // Opaque Client generated event identifier, used by DOTS client
       // to associate a mitigation status with the event triggering the
       // mitigation request.
       string eventid = 1;

       // Mitigation state
       bool enabled = 2;

       // Mitigation time-to-live (lifetime - (now - start))
       uint32 ttl = 3;

       // Dropped byte count
       uint64 bytes_dropped = 4;

       // Dropped bits per second
       uint64 bps_dropped = 5;

       // Dropped packet count
       uint64 pkts_dropped = 6;

       // Dropped packets per second
       uint64 pps_dropped = 7;

       // Blacklist enabled through data channel
       bool blacklist_enabled = 8;

       // Whitelist enabled through data channel
       bool whitelist_enabled = 9;

       // Filters enabled through data channel
       bool filters_enabled = 10;

       repeated google.protobuf.Any extensions = 999;
     }
```

Figure 19: DOTS Server Mitigation Status Schema

8.7. Server Error Schema

```
       syntax = "proto3";
       import "google/protobuf/any.proto";

       message DOTSServerError {
         enum ErrorCode {
           NOERROR = 0;
           INVALID_VALUE = 1;
           MITIGATION_UNAVAILABLE = 2;
           MITIGATION_CONFLICT = 3;
           MALFORMED_MESSAGE = 4;
         }
         ErrorCode code = 1;

         // Error details, returned as a blob
         google.protobuf.Any details = 2;
       }
```

Figure 20: DOTS Server Error Schema

9. References

9.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <http://www.rfc-editor.org/info/rfc3031>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-03 (work in progress), October 2016.

[I-D.hamilton-quic-transport-protocol]
Hamilton, R., Iyengar, J., Swett, I., and A. Wilk, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-hamilton-quic-transport-protocol-01 (work in progress), October 2016.

[PROTOBUF]
Google, Inc., "Protocol Buffers", 2016, <https://developers.google.com/protocol-buffers/>.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/˜fielding/pubs/dissertation/fielding_dissertation.pdf>.

9.2. Informative References

[RFC1518] Rekhter, Y. and T. Li, "An Architecture for IP Address Allocation with CIDR", RFC 1518, DOI 10.17487/RFC1518, September 1993, <http://www.rfc-editor.org/info/rfc1518>.

[RFC1519] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, DOI 10.17487/RFC1519, September 1993, <http://www.rfc-editor.org/info/rfc1519>.

[RFC4559] Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, DOI 10.17487/RFC4559, June 2006, <http://www.rfc-editor.org/info/rfc4559>.

[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, <http://www.rfc-editor.org/info/rfc5575>.

[RFC7617] Reschke, J., "The ’Basic’ HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <http://www.rfc-editor.org/info/rfc7617>.

[CLOUDSIGNAL]
Arbor Networks, Inc., "Cloud Signaling: A Faster, Automated Way to Mitigate DDoS Attacks", 2011, <https://www.arbornetworks.com/cloud-signaling-a-faster-automated-way-to-mitigate-ddos-attacks>.

[COMMUNITYFS]
Team Cymru, Inc., "Community FlowSpec", 2011, <https://www.cymru.com/jtk/misc/community-fs.html>.

[OPENHYBRID]
Verisign, Inc., "Verisign OpenHybrid", 2016, <http://www.verisign.com/en_US/security-services/ddos-protection/open-api/index.xhtml>.

[WISR] Arbor Networks, Inc., "Worldwide Infrastructure Security Report", 2016, <https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf>.

Authors’ Addresses

Nik Teague
Verisign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States

Email: nteague@verisign.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

Email: amortensen@arbor.net

DOTS
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2017

N. Teague
Verisign, Inc.
A. Mortensen
Arbor Networks, Inc.

February 15, 2017

DDoS Open Threat Signaling Protocol
draft-teague-dots-protocol-02

Abstract

This document describes Distributed-Denial-of-Service (DDoS) Open Threat Signaling (DOTS), a protocol for requesting and managing mitigation of DDoS attacks.

DOTS mitigation requests over the signal channel permit domains to signal the need for help fending off DDoS attacks, setting the scope and duration of the requested mitigation. Elements called DOTS servers field the signals for help, and enable defensive countermeasures to defend against the attack reported by the clients, reporting the status of the delegated defense to the requesting clients. DOTS clients additionally may use a reliable data channel to manage filters and black- and white-lists to restrict or allow traffic to the clients’ domains arbitrarily. The DOTS signal channel may operate over UDP [RFC0768] and if necessary TCP [RFC0793]. This revision discusses a transport-agnostic approach to this channel, focusing on the message exchanges and delegating transport specifics to other documents. Discussion of the reliable data channel may be found in [I-D.reddy-dots-data-channel].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Distributed-Denial-of-Service attack scale and frequency continues to increase year over year, and the trend shows no signs of abating [WISR]. In response to the DDoS attack trends, service providers and vendors have developed various approaches to sharing or delegating responsibility for defense, among them ad hoc service relationships, filtering through peering relationships [COMMUNITYFS], and proprietary solutions ([CLOUDSIGNAL], [OPENHYBRID]). Such hybrid approaches to DDoS defense have proven effective, but the heterogeneous methods employed to coordinate DDoS defenses across domain boundaries have necessarily limited their scope and effectiveness, as the mechanisms in one domain have no traction in another.

The DDoS Open Threat Signaling (DOTS) protocol provides a common mechanism to achieve the coordinated attack response previously restricted to custom or proprietary solutions. To meet the needs of network operators facing down modern DDoS attacks, DOTS itself is a hybrid protocol, consisting of a signal channel and a data channel. DOTS uses the signal channel, a lightweight and robust communication layer, to signal the need for mitigation regardless of network conditions, and uses the reliable data channel as vehicle for provisioning, configuration, telemetry, filter management, and other unsized or bulk data exchange.

The DOTS protocol is not intended as a replacement for such protocols as BGP Flow Specification [RFC5575] or as a general purpose DDoS countermeasure application programming interface (API), but rather as an advisory protocol enabling attack response coordination between DOTS agents. Any DOTS-enabled device or service is capable of triggering a request for help and shaping the scope and nature of that help, with the details of the actual mitigation left to the discretion of the operators of the attack mitigators. DOTS thereby permits all participating parties to manage their own attack defenses in the manner most appropriate for their own domains.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terms used to define entity relationships, transmitted data, and methods of communication are drawn from the terminology defined in [I-D.ietf-dots-requirements].

2. Architecture

The architecture in which the DOTS protocol operates is assumed to be derived from the architectural components and concepts described in [I-D.ietf-dots-architecture].

2.1. DOTS Agents

All protocol communication is between a DOTS client and a DOTS server. The logical agent termed a DOTS gateway is in practice a DOTS server placed back-to-back with a DOTS client. As discussed in [I-D.ietf-dots-architecture], any interface enabling the back-to-back DOTS server and client to act as a DOTS gateway is implementation-specific. This protocol is therefore concerned only with managing one or more bilateral relationships between DOTS clients and the DOTS servers, a signaling mode known as Direct Signaling in the DOTS architecture. This is shown in Figure 1 below:

```
+-----------+  signal channel  +-----------+
|           |<---------------->|           |
|DOTS client|                  |DOTS server|
|           |<================>|           |
+-----------+   data channel   +-----------+
```

Figure 1: DOTS protocol direct signaling

The DOTS architecture anticipates many-to-one and one-to-many deployments, in which multiple DOTS clients maintain distinct signaling sessions with a single DOTS server or a single DOTS client maintains distinct signaling sessions with multiple DOTS servers, as shown below in Figure 2:

```
       +----+      +----+      +----+
       | c1 |      | Sa |------| c2 |
       +----+      +----+      +----+
             \                /
              \              /
               \   +----+   /
                +--| Sb |--+
                   +----+

       DOTS        DOTS        DOTS
       client 1    servers     client 2
```

Figure 2: DOTS protocol direct signaling

DOTS server Sb has signaling sessions with DOTS clients c1 and c2. DOTS client c2 has signaling sessions with DOTS servers Sa and Sb. Except where explicitly defined in this protocol, all mechanisms to maintain multiple signaling sessions are left to the implementation.

3. DOTS Communication Channels

DOTS uses two channels, a signal channel and a data channel.

The signal channel is the minimal secure communication layer a DOTS client uses to request mitigation for resources under the administrative control the DOTS client; administrative control may be delegated.

The data channel offers DOTS clients a limited ability to manage configuration and attack filtering for requested mitigations. The data channel offers DOTS client operators the limited ability to adjust configuration and filtering for their mitigation requests.

This document describes the DOTS signal channel. The data channel is defined separately in [I-D.reddy-dots-data-channel].

4. Signal Channel

The purpose of the signaling channel is to convey DDoS mitigation request and status information between participating agents (client and server or gateway). Conditions during a DDoS attack are invariably hostile for connection-oriented protocols traversing affected paths. Mechanisms such as Happy Eyeballs [RFC6555] may be used to select a transport suitable for a given time and prevailing network conditions.

Implementations are required to support the DOTS signal channel over UDP as specified in [I-D.ietf-dots-requirements]. This document therefore assumes the the availability of a transport based upon UDP [RFC0768], but also defines the message exchanges agnostic of the underlying transport used to convey the signaling.

Key tenets of DOTS protocol design are low communication overhead and efficient message packing to increase the chances of successful transmission and receipt. Desirable side-effects of efficient packing are the removal of the possibility of fragmentation in addition to a message size that is friendly towards encapsulation (e.g via GRE [RFC2784] or MPLS [RFC3031]). Large UDP packets may also be treated adversely by middleboxes with restrictive policies or may fall foul of aggressive filtering.

In support of operational requirements for protocol efficiency, backward compatibility and extensibility in [I-D.ietf-dots-requirements], the signaling channel uses Protocol Buffers [PROTOBUF], also known as Protobufs, to define message schemas and serialize messages exchanged between DOTS agents.

The security requirements described in [I-D.ietf-dots-requirements] (peer mutual authentication, message confidentiality and integrity, and message replay protection are not discussed here. However, these qualities must be present in the transport over which the DOTS signal channel operates.

Key distribution in support of those security requirements to may be achieved via the data channel, via an online mechanism such as DANE [RFC6698], Enrollment over Secure Transport [RFC7030], or by out-of-band means. The key distribution mechanism is not specified in this document, but operators must take care to ensure the distribution provides the same level of security demanded by the signal channel itself.

4.1. Minimum Viable Information

DOTS is intended to be extensible and to evolve to meet the future needs in communicating as yet unknown threats. However, it must be able to convey the minimum information required for an upstream mitigation platform to successfully counter a DDoS attack. A client may have limited visibility into the full breadth of an attack and as such may not be well placed to provide useful telemetry. DDoS sources may or may not be spoofed and number in the millions. Once mitigation is active, the filtered traffic seen by the DOTS client (or elements informing the DOTS client operator’s decision to request mitigation) may not be representative of the ongoing attack. This provides challenges for the quality and usefulness of telemetry and mitigation/countermeasure stipulations and as such this type of information if conveyed can only be considered advisory.

In these instances the minimum viable information required for the majority of mitigations to be activated is that which pertains to the resource being targeted by the attack (host, prefix, protocol, port, URI etc.), per [I-D.ietf-dots-requirements] (OP-006). The DOTS requirements also identify a mitigation lifetime period (OP-005) and mitigation efficacy metric (OP-007). The former may be considered for inclusion in the minimum viable information set, however, the latter may only be relevant in updates. An explicit mitigation request/terminate flag is also required: a mitigation MUST be explicitly requested by a DOTS client operator. Finally, each message should include a message id or sequence number field as well as a field for the last received message id or sequence number. These may then be compared by the endpoints to assist in tracking state and/or identifying loss.

4.2. Signal Channel Messages

4.2.1. Messaging Overview

The signal channel requirements in [I-D.ietf-dots-requirements] demand a protocol capable of operating despite significant link congestion between DOTS agents. In an effort to maximize the probability of signal delivery between DOTS agents, all DOTS signal channel messages in the DOTS protocol are conceptually unidirectional heartbeats sent between DOTS agents.

The result is a form of scheduled messaging between DOTS agents, in contrast to a conventional request-response model. Once a signal channel is established, a DOTS client begins sending unidirectional heartbeats to the DOTS server, separated by the configured session heartbeat interval (see Section 4.3.1 below). To request mitigation, a DOTS client merely adds the requested mitigation parameters to its heartbeat, and maintains that request until a heartbeat from the DOTS server indicates receipt of that mitigation request. The DOTS server likewise sends its unidirectional heartbeat on the schedule interval, augmenting the content of the heartbeat with mitigation request status.

A simple exchange between DOTS agents with an established signaling session is shown below in Figure 3.

```
   Client                           Server
     |                                 |  // Active signaling session
     |<==== Signal Channel Active ====>|  // with heartbeat interval of
     |                                 |  // 15 seconds, +/- jitter.
     |                                 |
     |----------Heartbeat------------->|  // Client heartbeat
     |                                 |
     |<---------Heartbeat--------------|  // Server heartbeat
     |                                 |
     \                                 \  // 15 seconds elapse, during
     /                                 /  // which client adds request
     \                                 \  // for mitigation.
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat now has
     |          + MitigationRequest    |  // mitigation request params.
     |                                 |
     |<---------HeartBeat--------------|  // Server heartbeat now has
     |          + MitigationPending    |  // pending mitigation request
     |                                 |  // status.
     |                                 |
     \                                 \
     /                                 /  // 15s elapse, +/- jitter.
     \                                 \
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat maintains
     |          + MitigationRequest    |  // request as it waits for
     |                                 |  // mitigation to begin.
     |                                 |
     |     X----HeartBeat--------------|  // Mitigation enabled, but
     |          + MitigationRunning    |  // server heartbeat is dropped
     |                                 |
     \                                 \
     /                                 /  // 15s elapse, +/- jitter.
     \                                 \
     |                                 |
     |----------HeartBeat------------->|  // Client heartbeat maintains
     |          + MitigationRequest    |  // request as it waits for
     |                                 |  // mitigation to begin.
     |                                 |
     |<---------HeartBeat--------------|  // Server heartbeat, status of
     |          + MitigationRunning    |  // running mitigation
     |                                 |  // delivered to client.
```

Figure 3: Signaling Session with Mitigation Request

All messages are variations of this scheduled heartbeat model. See Section 4.3 below for detailed discussion of the message exchanges between DOTS agents.

4.2.2. Message Definition and Serialization

The DOTS protocol signal channel uses Protobufs [PROTOBUF] as an interface definition language (IDL) for signal channel messaging between DOTS agent, reducing the number of discrete messages to just a single message superset per direction, with function defined by the chosen fields contained within the message.

Protobufs schemas are used to define the messages sent in either direction, from which code may be generated for specific language implementations of DOTS. As Protobufs serialization relies on numbered fields, signal channel messaging permits the introduction of new numbered fields arbitrarily, adding the requisite extensibility to the protocol while retaining backward compatibility. Future revisions of or extensions to the protocol may use the data channel to provide a mechanism by which schema updates or expansions may be communicated during provisioning/session establishment.

4.2.3. Client Message Fields

Only the seqno and last_svr_seqno fields are required in every message, as they are the minimum required for the heartbeat. Subsets of the other fields are used to convey a given signal message type.

The fields in the DOTS client signal channel message have the following functions:

seqno: a client-generated sequence number unique to the message. The client increments the seqno value by one for each message sent over the signal channel.

last_svr_seqno: the sequence number of the last message received from the server, provided to the server as a simple way to detect lost messages.

mitigations: a list of mitigations requested or withdrawn by the client. The mitigation schema fields are described below.

active: indicates a request for a list of active mitigations and their detail that are current on the DOTS server.

ping: an operator initiated heartbeat like message which will elicit a response from the DOTS server. This may be used to prove bi-directional communications on an ad-hoc basis. For example, a DOTS ping may be used to prove keying material on the DOTS client is valid and may be used to establish signaling sessions with the DOTS server.

extensions: these fields may be used to communicate implementation specific details. An example would be the dissemination of filters between DOTS client and DOTS server.

4.2.3.1. Client Message Schema

The DOTS client message schema is detailed in Section 7.1.

4.2.4. Mitigation Request Fields

The fields in a mitigation request are as follows:

eventid: an opaque client generated identifier that distinguishes a unique event or incident. May be used by the client as a reference to the specific event triggering a mitigation request, or for other implementation-specific purposes.

requested: signals the need for mitigation to the DOTS server. If true, the DOTS client is requesting mitigation for the provided scope. If false, the DOTS client is indicating it does not require mitigation, and the DOTS server MUST cease the mitigation for the provided scope.

scope: the scope of the mitigation requested, which may be any of the types described in [I-D.ietf-dots-requirements], such as Classless Internet Domain Routing (CIDR) [RFC1518],[RFC1519] prefixes, DNS names, or aliases defined by the DOTS client operator through the data channel.

lifetime: the lifetime in seconds a mitigation request should be considered valid.

efficacy: a metric to convey to a DOTS server the perceived efficacy of an active mitigation, per operational requirements in [I-D.ietf-dots-requirements]. The mitigation efficacy is represented as a floating point value between 0 and 1, with smaller values indicating lesser efficacy, and larger greater efficacy.

extensions: these fields may be used to provide implementation-specific mitigation details.

4.2.4.1. Mitigation Request Schema

The DOTS client message schema is detailed in Section 7.2.

4.2.5. DOTS Server Message Fields

DOTS server messages use a subset of the available fields to convey the given signal type, including additional relevant fields as necessary. The only fields which may be common to all signals are seqno and last_client_seqno which may be used to detect message loss or out-of-order delivery. When conveying mitigation information, the server schema may bundle multiple mitigation status datasets into a single message, provided this does not violate the required sub-MTU message size [I-D.ietf-dots-requirements].

The fields in the DOTS server signal channel message schema have the following functions:

seqno: a server generated sequence number unique to the message.

last_cli_seqno: the seqno of the last message received from the client.

ping: an operator-initiated heartbeat like message which will elicit a response from the DOTS client. This may be used to prove bi-directional communications on an ad-hoc basis.

error: details of an error caused by a DOTS client request.

redirect: Populated with the details of the redirection target DOTS server, if the DOTS server is redirecting the DOTS client to another DOTS server.

mitigations: a list containing the status of mitigations requested by the DOTS client. The fields in the mitigation status schema are described below.

extensions: these fields may be used to communicate implementation specific details. An example would be the communication of DNS mitigation vip to the DOTS client by the DOTS server.

4.2.5.1. DOTS Server Message Schema

The server message schema is detailed in Section 7.4.

4.2.6. Server Errors

If a DOTS client message cannot be processed by the DOTS server, or for any other reason causes an error, the DOTS server MUST populate the error field in any response to the message causing the error. As the error response itself may be lost, a DOTS client may continue sending problematic messages regardless of the DOTS server’s error notifications. DOTS server implementations MAY terminate the signaling session after client-triggered errors exceed a threshold during a time period equivalent to three times the session heartbeat interval.

The DOTS client message triggering the error condition is indicated in the last_client_seqno value of the DOTS server message containing the error.

Error codes MUST be one of the following types:

NOERROR: Indicates the DOTS server has detected no error resulting from a DOTS client message. Implementations MAY omit the error field entirely when no error condition is present. This value is included in the schema largely to adhere to the convention that an error status of 0 indicates success.

INVALID_VALUE: Indicates the DOTS client included an invalid value for a field in the client message most recently received from the client. The DOTS server SHOULD include specifics of the invalid value in the details field of the error.

MITIGATION_UNAVAILABLE: Indicates the DOTS server is unable to provide mitigation in response to a mitigation request from the DOTS client.

MITIGATION_CONFLICT: Indicates a mitigation request conflicts with an existing mitigation from the client. The DOTS server SHOULD populate the error details field with the status information of the mitigation conflicting with the requested mitigation.

MALFORMED_MESSAGE: Indicates the DOTS client message is malformed and cannot be processed.

4.2.6.1. Server Error Fields

code: a numeric code categorizing the error type detected by the DOTS server.

details: specific information about the reason for the detected error.

4.2.6.2. Server Error Schema

The server error schema is detailed in Section 7.7.

4.2.7. Server Mitigation Status Fields

The DOTS server message contains zero or more mitigation status messages, the fields of which have the following functions:

eventid: an opaque client generated identifier that distinguishes a unique event or incident.

ttl: the remaining lifetime of the mitigation, in seconds.

bytes_dropped: the total dropped byte count for the mitigation associated with eventid.

bps_dropped: the dropped bytes per second for the mitigation associated with eventid. This value is expected to be calculated by the mitigator, and as such is implementation-specific.

pkts_dropped: the total dropped packet count for the mitigation associated with eventid..

pps_dropped: the dropped packets per second for the mitigation associated with eventid. This value is expected to be calculated by the mitigator, and as such is implementation-specific.

blacklist_enabled: Indicates whether a blacklist of prohibited traffic sources is enabled for the mitigation associated with eventid. The blacklist is managed through the data channel.

whitelist_enabled: Indicates whether a whitelist of sources from which traffic must always be allowed is enabled. The whitelist is managed through the data channel.

filters_enabled: Indicates whether client-specified traffic filters are enabled for the mitigation associated with eventid.

4.3. Interactions

4.3.1. Session Initialization

Signaling sessions are initiated by the DOTS client. Session initialization begins when the DOTS client connects to the DOTS server port, 4646. After connecting, the DOTS client establishes the channel security context, including all necessary cryptographic exchanges between the two DOTS agents.

This signal channel specification is transport-agnostic, and delegates the details of transport, including transport security, to transport-specific documents. Regardless of transport, DOTS implementations nonetheless MUST provide signal channel security meeting the requirements in [I-D.ietf-dots-requirements].

Once the signal channel security context is established, the DOTS client sends a channel initialization message to the DOTS server, optionally including signaling session configuration values; if the session configuration values are excluded, defaults MUST be used for the signaling session. An example initialization message setting the acceptable signal loss and heartbeat interval for the signaling sessions is described in Figure 4 below:

```
    message DOTSClientMessage {
      1 (seqno) = %;
      2 (last_svr_seqno) = %;
      6 (config) = {
        1 (loss_limit) = %;
        3 (heartbeat_interval) = %;
      };
    }
```

Figure 4: Signal Channel Initialization Message

The DOTS server MUST respond immediately by sending a heartbeat (see Section 4.3.2 below) to the DOTS client. The signal channel is active when the DOTS client receives a heartbeat from the DOTS server with a last_client_seqno of a signal channel initialization message. Both DOTS agents MUST begin sending heartbeats on the interval for the signaling session once the session is active.

The following example assumes a DOTS implementation using UDP as the transport and DTLS1.2 [RFC6347]. In Figure 5 below, the DOTS client uses the default values for acceptable signal loss, maximum mitigation lifetime, and heartbeat interval. The initial DOTS server heartbeat is lost, so the DOTS client sends another channel initialization message after waiting for the minimum heartbeat interval defined below in Section 4.3.2:

```
   Client                            Server
     |                                 |
     |<- - - - DTLS1.2 handshake - - ->|  // Server listens on UDP:4646.
     |                                 |  // Client initiates DTLS
     |                                 |  // handshake.
     |                                 |
     |----------ChannelInit----------->|  // Client sends signal
     |          seqno = 1              |  // channel init message
     |          last_svr_seqno = 0     |
     |                                 |
     |     X----HeartBeat--------------|  // Server immediately sends
     |          seqno = 1              |  // heartbeat reply, which
     |          last_client_seqno = 1  |  // is lost.
     |                                 |
     \                                 \  // Session heartbeat interval
     /                                 /  // elapses.
     \                                 \
     |----------ChannelInit----------->|  // Client retries signal
     |          seqno = 2              |  // channel init message.
     |          last_svr_seqno = 0     |
     |                                 |
     |<---------HeartBeat--------------|  // Server immediately sends
     |          seqno = 2              |  // heartbeat reply
     |          last_client_seqno = 2  |
     |                                 |
     |<==== Signal Channel Active ====>|
```

Figure 5: Signal Channel Initialization

4.3.1.1. Session Initialization Error Handling

If the DOTS client specifies invalid values for the signal channel configuration, the DOTS server replies with an error, and may ultimately terminate the connection if the client fails to correct the invalid values, as described in [I-D.ietf-dots-architecture].

4.3.1.2. Mis-Sequencing

In the event that the DOTS agent receives messages containing invalid seqno, last_client_seqno or last_svr_seqno these should be discarded and ignored.

4.3.2. Heartbeat

The most common message exchanged between a DOTS client and a DOTS server is a heartbeat (OP-002 [I-D.ietf-dots-requirements]), which maintains and monitors the health of the DOTS session. This is achieved with simple, loosely-coupled bi-directional messages containing the sending DOTS agent’s message sequence number and the sequence number the sending DOTS agent last received from its peer. Due to the stress volumetric DDoS impose upon a network, a degree of loss during attacks is to be expected. Message loss tolerance may be set on signal channel establishment.

The default heartbeat interval is 20 seconds, plus or minus a number of milliseconds between 50 and 2000. The number of milliseconds MUST be randomized in order to introduce jitter into the heartbeat interval, as recommended by [RFC5405]. The default interval is derived from the recommendations in [RFC5405] regarding middlebox traversal, to maintain NAT bindings in the path between DOTS agents.

The interval between heartbeats is may also be set by the client when establishing the signal channel. The minimum heartbeat interval is 15 seconds, plus the random number of milliseconds as described above. The maximum heartbeat interval is 120 seconds (two minutes), minus the random number of milliseconds described above.

Heartbeats are loosely-coupled, meaning each DOTS agent in a bilateral signaling session sends DOTS heartbeats on the specified interval, but asynchronously, without acknowledgement. Each DOTS agent tracks heartbeats received from its peer, and includes the sequence number of the last heartbeat received from the peer agent in the next heartbeat sent, as shown in Figure 6:

```
     Client                            Server
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 1              |
       |          last_svr_seqno = 0     |
       |                                 |
       |<---------HeartBeat--------------|  // Server heartbeat
       |          seqno = 1              |
       |          last_client_seqno = 1  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 2              |
       |          last_svr_seqno = 1     |
       |                                 |
       |      X---HeartBeat--------------|  // Server heartbeat lost
       |          seqno = 2              |
       |          last_client_seqno = 2  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat,
       |          seqno = 3              |  // last_svr_seqno remains 1,
       |          last_svr_seqno = 1     |  // indicating lost heartbeat
       |                                 |
       |<---------HeartBeat--------------|  // Server heartbeat resumes
       |          seqno = 3              |
       |          last_client_seqno = 3  |
       |                                 |
       |----------HeartBeat------------->|  // Client heartbeat
       |          seqno = 4              |
       |          last_svr_seqno = 3     |
       |                                 |
```

Figure 6: Heartbeats Between DOTS agents

The DOTS client heartbeat has the following format:

```
message DOTSClientMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
}
```

The DOTS server heartbeat is identical aside from the schema type:

```
message DOTSServerMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
}
```

Should the number of signals lost exceed the acceptable lossiness value for the signaling session, the agent detecting the signal loss may consider the signaling session lost. The default value for acceptable signal loss is 9, which, when coupled with the default heartbeat interval, amounts to lack of heartbeat from the peer DOTS agent for 180 seconds (three minutes).

4.3.2.1. Ping

There may be cases where a DOTS client or server operator wishes to trigger an immediate heartbeat response in order to validate bi-directional communication (e.g. during provisioning). This ad-hoc triggering may be achieved by setting the ping field set to TRUE. When DOTS agent receives a message on the signal channel with the ping field set to TRUE, it MUST immediately send heartbeat back to the ping sender. A ping reply MUST consist of only the senders sequence number and the sequence number of the received ping. [[EDITOR’S NOTE: rate limiting of pings required?]]

A ping is identical to a standard heartbeat, but with the ping field included and set to true:

```
message DOTSClientMessage {
  1 (seqno) = %;
  2 (last_svr_seqno) = %;
  5 (ping) = true;
}
```

4.3.3. Mitigation Request Handling

The mitigation request is the crux of the DOTS protocol, and is comprised of the minimum viable information described in Section 4.1. The request may be augmented with additional implementation specific extensions where these do not result in undue packet bloat. The DOTS client may send repeated requests until it receives a suitable response from the DOTS server by which it may interpret successful receipt.

```
         message DOTSClientMessage {
           1 (seqno) = %;
           2 (last_svr_seqno) = %;
           3 (mitigations) = [
             {
               1 (eventid) = %;
               2 (requested) = %;
               3 (scope) = %;
               4 (lifetime) = %;
             }
           ];
         }
```

The DOTS server is expected to respond to confirm that it has accepted and or rejected the mitigation request. Upon receipt of the response the DOTS client should cease sending additional initial requests for the same eventid. If these do not cease then the server may assume that the response was possibly lost and should resend accordingly. Acceptance status is communicated by the DOTS server replying with the corresponding eventid and the enabled field set to 1 for acceptance and 0 for rejection. A rejection by the DOTS server should be accompanied with an extension field detailing succinctly the reason (e.g. out of contract, conflict, maintenance etc.).

```
        message DOTSServerMessage {
          1 (seqno) = %;
          2 (last_cli_seqno) = %;
          4 (mitigations) = [
            {
              1 (eventid) = %;
              2 (enabled) = true; // Mitigation request accepted
            }
          ]
        }
```

After a period of time the mitigation request may expire and the DOTS server may end the mitigation. Alternately, the DOTS client may explicitly terminate the active mitigation by sending a message to the server that contains a mitigation value with the eventid and that has the requested field set to false, as shown below:

```
         message DOTSClientMessage {
           1 (seqno) = %;
           2 (last_svr_seqno) = %;
           3 (mitigations) = [
             {
               1 (eventid) = %;
               2 (requested) = false; // Terminate mitigation
             }
           ];
         }
```

The server must explicitly acknowledge the termination with a response message with the enabled field now set to false:

```
         message DOTSServerMessage {
           1 (seqno) = %;
           2 (last_cli_seqno) = %;
           6 (mitigations) = [
             {
               1 (eventid) = %;
               2 (enabled) = false; // Mitigation terminated
             }
           ];
         }
```

The life cycle of a DOTS mitigation request resembles the following:

```
          Client                        Server
            |                              |
            |---Request(M=true)----------->|  // Mitigation request
            |                              |
            |<---------MitigationActive----|  // Server acceptance
            |                              |
            |< - - - - MitigationFeedback -|
            |                              |
            |---Terminate(M=false)-------->|  // Mitigation termination
            |                              |
            |<---------MitigationEnded-----|  // Server termination ack
```

4.3.4. Ancillary Messages

In addition to the basic interaction, additional messages may be exchanged throughout the lifetime of the mitigation. The following message types are defined to provide requisite information between DOTS agents during an active signaling session.

4.3.4.1. Mitigation Feedback

The DOTS server MUST update the client with current mitigation status. This MUST include the eventid, and SHOULD include available dropped attack traffic statistics provided by the mitigator. A DOTS server MAY provide feedback for more than one mitigation in a single message, provided the resulting message meets the sub-MTU size requirements in [I-D.ietf-dots-requirements].

The DOTS client SHOULD use the feedback from the DOTS server when deciding to update or terminate a mitigation request. For example, if the DOTS client learns from DOTS server mitigation feedback that the dropped_pps rate is low, the DOTS client might decide to terminate upstream mitigation and handle the attack locally.

A mitigation feedback message from the DOTS server would resemble the following format, assuming an active mitigation request from the DOTS client:

```
message DOTSServerMessage {
  1 (seqno) = %;
  2 (last_client_seqno) = %;
  6 (mitigations) = [
    {
      1 (eventid) = %;
      2 (enabled) = %;
      3 (ttl) = %;
      4 (bytes_dropped) = %;
      5 (bps_dropped) = %;
      6 (pkts_dropped) = %;
      7 (pps_dropped) = %;
      10 (filters_enabled) = true;
    },
  ];
}
```

4.3.4.2. Mitigation Lifetime Update

The DOTS client may wish to update the mitigation during its lifetime. Updates may be to alter the lifetime to extend the mitigation, or an update may communicate the perceived efficacy of the mitigation. The former may be as a result of the DOTS sever feedback which may suggest that an attack shows no sign of abating. The latter may be to notify the DOTS server whether the countermeasures deployed are perceived as effective or not.

A DOTS client may update the lifetime of multiple mitigations in a single request as long as the message size meets the sub-MTU requirement per [I-D.ietf-dots-requirements]. The lifetime update message has the following format:

```
       message DOTSClientMessage {
         1 (seqno) = %;
         2 (last_svr_seqno) = %;
         3 (mitigations) = [
           {
             1 (eventid) = %;
             2 (requested) = true;
             4 (lifetime) = %;
           }
         ];
       }
```

Upon receipt of the mitigation lifetime update, the DOTS server replace the current mitigation expiration time with the new value. The updated lifetime MUST be visible in the ttl field in subsequent mitigation feedback messages. When updating a mitigation lifetime, the DOTS client SHOULD continue sending the lifetime update request at the heartbeat interval until the DOTS server’s mitigation feedback shows an updated ttl for the updated mitigation.

4.3.4.3. Mitigation Efficacy Updates

When a mitigation is active, a DOTS client MUST periodically communicate the locally perceived efficacy of the mitigation to the DOTS server. This gives the DOTS server a rough sense of whether the DOTS client perceives the mitigator’s deployed countermeasures as effective. The efficacy update update message has the following format:

```
      message DOTSClientMessage {
        1 (seqno) = %;
        2 (last_svr_seqno) = %;
        3 (mitigations) = [
          {
            1 (eventid) = %;
            6 (efficacy) = %;
          }
        ];
      }
```

The DOTS server SHOULD consider the efficacy update an indication of the effectiveness of any ongoing mitigations related to the eventid provided by the DOTS client. The DOTS server nonetheless MAY treat any efficacy update from the client as advisory, and is under no obligation to alter the mitigation strategy in response.

5. IANA Considerations

The DOTS protocol requires a well-known port to which DOTS client messages are sent. The DOTS server listens on the well-known port for client messages. The DOTS client binds to an ephemeral port per Section 4.3.1 above. This document selects port 4646 (the ASCII decimal values for "..": "DOTS") as the well-known port.

6. Security Considerations

The DOTS protocol controls mitigation request and withdrawal and as such care must be taken to protect against concerns outlined in the security considerations of [I-D.ietf-dots-architecture].

7. Appendix A: Message Schemas

7.1. DOTS Client Message Schema

```
syntax = "proto3";
import "google/protobuf/any.proto";

message DOTSClientMessage {
  // Client generated sequence number
  uint64 seqno = 1;

  // Sequence number of last received server message
  uint64 last_svr_seqno = 2;

  repeated DOTSMitigation mitigations = 3;

  // Request active mitigation list from server
  bool active = 4;

  // Ping request (operator initiated)
  bool ping = 5;

  DOTSSessionConfig config = 6;

  repeated google.protobuf.Any extensions = 999;
}
```

Figure 7: DOTS Client Message Schema

7.2. Mitigation Request Schema

```
       message DOTSMitigation {
         // Opaque client-generated event identifier
         string eventid = 1;

         // Toggle mitigation for the above scope
         bool requested = 2;

         // Mitigation scope as described in I-D.ietf-dots-requirements
         string scope = 3;

         // Lifetime of the requested mitigation.
         uint32 lifetime = 4;

         // Mitigation efficacy score as a float value between 0 and 1
         float efficacy = 5;

         repeated google.protobuf.Any extensions = 999;
       }
```

Figure 8: DOTS Client Mitigation Request Schema

7.3. Session Configuration Schema

```
         // Per session configuration sent on signaling session init
         message DOTSSessionConfig {
           // Acceptable signal loss
           uint32 loss_limit = 1;

           // Maximum mitigation lifetime in seconds
           uint32 lifetime_max = 2;

           // Heartbeat interval in milliseconds
           uint32 heartbeat_interval = 3;
         }
```

Figure 9: DOTS Session Conifiguration Schema

7.4. DOTS Server Message Schema

```
       syntax = "proto3";
       import "google/protobuf/any.proto";

       message DOTSServerMessage {
         // Server generated sequence number
         uint64 seqno = 1;

         // Sequence number of last received Client message
         uint64 last_client_seqno = 2;

         // Request immediate heartbeat response from client.
         bool ping = 3;

         // Server error details, if available
         DOTSServerError error = 4;

         DOTSRedirect redirect = 5;

         // Mitigation data, limited by MTU
         repeated DOTSMitigationStatus mitigations = 6;
       }
```

Figure 10: DOTS Server Message Schema

7.5. DOTS Redirect Schema

```
       message DOTSRedirect {
         // Redirection target DOTS server address
         string target = 1;

         // Address family of redirection target
         enum RedirectionTargetType {
           DNSNAME = 0;
           IPV4 = 4;
           IPV6 = 6;
         }
         RedirectionTargetType target_type = 2;

         // Port on which to contact redirection target.
         // XXX Protobufs has no uint16 type, implementations
         // will need to sanity check.
         uint32 port = 3;
       }
```

7.6. DOTS Mitigation Status Schema

```
syntax = "proto3";
import "google/protobuf/any.proto";

message DOTSMitigationStatus {
  // Opaque Client generated event identifier, used by DOTS client
  // to associate a mitigation status with the event triggering the
  // mitigation request.
  string eventid = 1;

  // Mitigation state
  bool enabled = 2;

  // Mitigation time-to-live (lifetime - (now - start))
  uint32 ttl = 3;

  // Dropped byte count
  uint64 bytes_dropped = 4;

  // Dropped bits per second
  uint64 bps_dropped = 5;

  // Dropped packet count
  uint64 pkts_dropped = 6;

  // Dropped packets per second
  uint64 pps_dropped = 7;

  // Blacklist enabled through data channel
  bool blacklist_enabled = 8;

  // Whitelist enabled through data channel
  bool whitelist_enabled = 9;

  // Filters enabled through data channel
  bool filters_enabled = 10;

  repeated google.protobuf.Any extensions = 999;
}
```

Figure 11: DOTS Server Mitigation Status Schema

7.7. Server Error Schema

```
syntax = "proto3";
import "google/protobuf/any.proto";

message DOTSServerError {
  enum ErrorCode {
    NOERROR = 0;
    INVALID_VALUE = 1;
    MITIGATION_UNAVAILABLE = 2;
    MITIGATION_CONFLICT = 3;
    MALFORMED_MESSAGE = 4;
  }
  ErrorCode code = 1;

  // Error details, returned as a blob
  google.protobuf.Any details = 2;
}
```

Figure 12: DOTS Server Error Schema

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

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

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <http://www.rfc-editor.org/info/rfc3031>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[I-D.ietf-dots-architecture]
Mortensen, A., Andreasen, F., Reddy, T., christopher_gray3@cable.comcast.com, c., Compton, R., and N. Teague, "Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture", draft-ietf-dots-architecture-01 (work in progress), October 2016.

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed Denial of Service (DDoS) Open Threat Signaling Requirements", draft-ietf-dots-requirements-03 (work in progress), October 2016.

[PROTOBUF]
Google, Inc., "Protocol Buffers", 2016, <https://developers.google.com/protocol-buffers/>.

8.2. Informative References

[I-D.reddy-dots-data-channel]
Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., and P. Patil, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel", draft-reddy-dots-data-channel-03 (work in progress), December 2016.

[RFC1518] Rekhter, Y. and T. Li, "An Architecture for IP Address Allocation with CIDR", RFC 1518, DOI 10.17487/RFC1518, September 1993, <http://www.rfc-editor.org/info/rfc1518>.

[RFC1519] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, DOI 10.17487/RFC1519, September 1993, <http://www.rfc-editor.org/info/rfc1519>.

[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, <http://www.rfc-editor.org/info/rfc5575>.

[CLOUDSIGNAL] Arbor Networks, Inc., "Cloud Signaling: A Faster, Automated Way to Mitigate DDoS Attacks", 2011, <https://www.arbornetworks.com/cloud-signaling-a-faster-automated-way-to-mitigate-ddos-attacks>.

[COMMUNITYFS] Team Cymru, Inc., "Community FlowSpec", 2011, <https://www.cymru.com/jtk/misc/community-fs.html>.

[OPENHYBRID] Verisign, Inc., "Verisign OpenHybrid", 2016, <http://www.verisign.com/en_US/security-services/ddos-protection/open-api/index.xhtml>.

[WISR] Arbor Networks, Inc., "Worldwide Infrastructure Security Report", 2016, <https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf>.

Authors’ Addresses

Nik Teague
Verisign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States

Email: nteague@verisign.com

Andrew Mortensen
Arbor Networks, Inc.
2727 S. State St
Ann Arbor, MI 48104
United States

Email: amortensen@arbor.net