DOTS
Internet-Draft
Intended status: Standards Track
Expires: August 12, 2016

T. Reddy
D. Wing
P. Patil
M. Geller
Cisco
M. Boucadair
France Telecom
R. Moskowitz
HTT Consulting
February 9, 2016

Co-operative DDoS Mitigation
draft-reddy-dots-transport-02

Abstract

This document discusses mechanisms that a DOTS client can use, when it detects a potential Distributed Denial-of-Service (DDoS) attack, to signal that the DOTS client is under an attack or request an upstream DOTS server to perform inbound filtering in its ingress routers for traffic that the DOTS client wishes to drop. The DOTS server can then undertake appropriate actions (including, blackhole, drop, rate-limit, or add to watch list) on the suspect traffic to the DOTS client, thus reducing the effectiveness of the attack.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 12, 2016.

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, etc. The reader may refer, for example, to [REPORT] that reports the following:

- o Very large DDoS attacks above the 100 Gbps threshold are experienced.

- o DDoS attacks against customers remain the number one operational threat for service providers, with DDoS attacks against infrastructures being the top concern for 2014.

- o Over 60% of service providers are seeing increased demand for DDoS detection and mitigation services from their customers (2014), with just over one-third seeing the same demand as in 2013.

In a lot of cases, it may not be possible for an enterprise to determine the cause for an attack, but instead just realize that certain resources seem to be under attack. The document proposes that, in such cases, the DOTS client just inform the DOTS server that the enterprise is under a potential attack and that the DOTS server monitor traffic to the enterprise to mitigate any possible attack. This document also describes a means for an enterprise, which act as DOTS clients, to dynamically inform its DOTS server of the IP addresses or prefixes that are causing DDoS. A DOTS server can use this information to discard flows from such IP addresses reaching the customer network.

The proposed mechanism can also be used between applications from various vendors that are deployed within the same network, some of them are responsible for monitoring and detecting attacks while others are responsible for enforcing policies on appropriate network elements. This cooperations contributes to a ensure a highly automated network that is also robust, reliable and secure. The advantage of the proposed mechanism is that the DOTS server can provide protection to the DOTS client from bandwidth-saturating DDoS traffic.

How a DOTS server determines which network elements should be modified to install appropriate filtering rules is out of scope. A variety of mechanisms and protocols (including NETCONF) may be considered to exchange information through a communication interface between the server and these underlying elements; the selection of appropriate mechanisms and protocols to be invoked for that

interfaces is deployment-specific.

Terminology and protocol requirements for co-operative DDoS mitigation are obtained from [I-D.ietf-dots-requirements].

2. Notational Conventions

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].

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 an application uses these resources to offer its intended task in the most efficient fashion. However, an attacker may be able to prevent the application from performing its intended task by causing the application to exhaust the finite supply of a specific resource.

TCP DDoS SYN-flood is a memory-exhaustion attack on the victim and ACK-flood is a CPU exhaustion attack on the victim. 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, if a network resource detects a potential DDoS attack from a set of IP addresses, the network resource (DOTS client) informs its servicing router (DOTS relay) of all suspect IP addresses that need to be blocked or black-listed for further investigation. DOTS client could also specify protocols and ports in the black-list rule. That DOTS relay in-turn propagates the black-listed IP addresses to the DOTS server and the DOTS server blocks traffic from these IP addresses to the DOTS client thus

reducing the effectiveness of the attack. The DOTS client periodically queries the DOTS server to check the counters mitigating the attack. If the DOTS client receives response that the counters have not incremented then it can instruct the black-list rules to be removed. If a blacklisted IPv4 address is shared by multiple subscribers then the side effect of applying the black-list rule will be that traffic from non-attackers will also be blocked by the access network.

If a DOTS client cannot determine the IP address(s) that are causing the attack, but is under an attack nonetheless, the DOTS client can just notify the DOTS server that it is under a potential attack and request that the DOTS server take precautionary measures to mitigate the attack.

4. Happy Eyeballs

If a DOTS server IPv4 path is working, but the DOTS server's IPv6 path is not working, a dual-stack DOTS client can experience significant connection delay compared to an IPv4-only DOTS client. The other problem is that if a middle box between the DOTS client and server is configured to block UDP, DOTS client will fail to establish DTLS session [RFC6347] with the DOTS server and will have to fall back to TLS over TCP [RFC5246] incurring significant connection delay. [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 MUST try connecting to the DOTS server using both IPv6 and IPv4, and MUST 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 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).

TBD: How does the DOTS client discover the DOTS server (use DNS-SD) ?

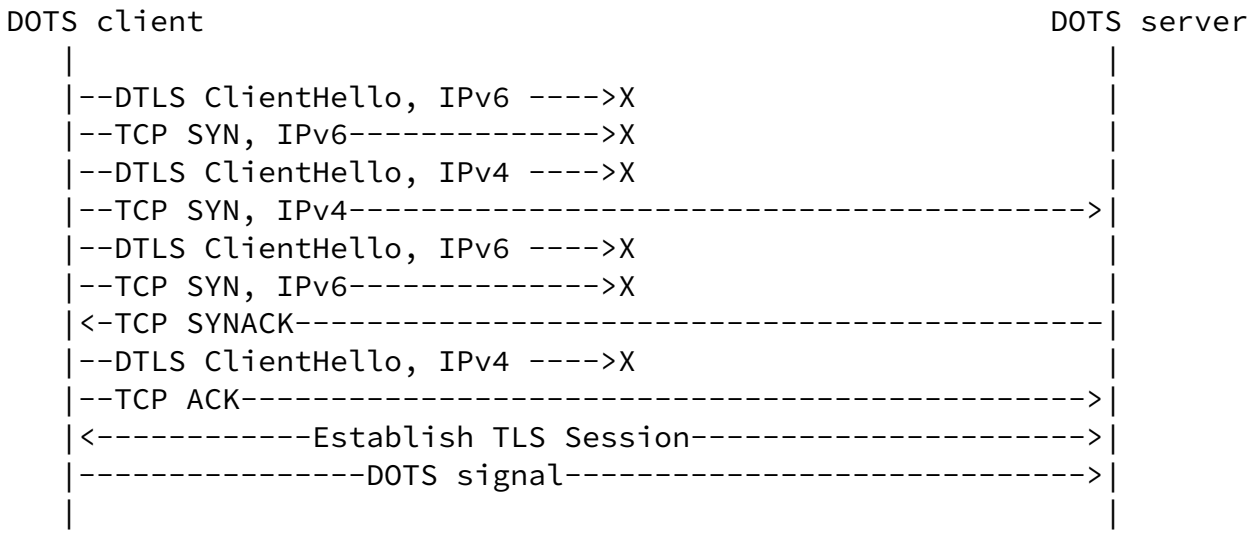

Figure 1: Happy Eyeballs

In the diagram above, the DOTS client sends two TCP SYNs and two DTLS ClientHello messages at the same time over IPv6 and IPv4. In the diagram, 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.

DOTS client and server can also use the following techniques to reduce delay to convey DOTS signal:

- DOTS client can use (D)TLS session resumption without server-side state [RFC5077] to resume session and convey the DOTS signal.

- TLS False Start [I-D.ietf-tls-falsestart] which reduces round-trips by allowing the TLS second flight of messages (ChangeCipherSpec) to also contain the DOTS signal.

- Cached Information Extension [I-D.ietf-tls-cached-info] 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.

- While the communication to the DOTS server is quiescent, the DOTS client may want to probe the server to ensure it has maintained cryptographic state. 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 server.

  * A DTLS heartbeat [RFC6520] verifies the server still has DTLS state by returning a DTLS message. If the server has lost state, it returns a DTLS Alert. Upon receipt of an un-authenicated DTLS alert, the DTLS client validates the Alert is within the replay window (Section 4.1.2.6 of [RFC6347]). It is difficult for the DTLS client to validate the DTLS alert was generated by the DTLS server in response to a request or was generated by an on- or off-path attacker. Thus, upon receipt of an in-window DTLS Alert, the client SHOULD continue re-transmitting the DTLS packet (in the event the Alert was spoofed), and at the same time it SHOULD initiate DTLS session resumption.

  * TLS runs over TCP, so a simple probe is a 0-length TCP packet (a "window probe"). This verifies the TCP connection is still working, which is also sufficient to prove the server has retained TLS state, because if the server loses TLS state it abandons the TCP connection. If the server has lost state, a TCP RST is returned immediately.

5. Protocol for Signal Channel: HTTP REST

A DOTS client can use RESTful APIs discussed in this section to signal/inform a DOTS server of an attack or any desired IP filtering rules.

5.1. Mitigation service request

The following APIs define the means to convey an DOTS signal from a DOTS client to a DOTS server.

5.1.1. Convey DOTS signal

An HTTP POST request will be used to convey DOTS signal to the DOTS server.

```
POST {scheme}://{host}:{port}/.well-known/{version}/{URI suffix for DOTS sign
Accept: application/json
Content-type: application/json
{
   "policy-id": number,
   "target-ip": string,
   "target-port": string,
   "target-protocol": string,
 }
```

Figure 2: POST to convey DOTS signal

The header fields are described below.

policy-id: Identifier of the policy represented using a number. 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 client and used as an opaque value by the server. This document does not make any assumption about how this identifier is generated. This is an mandatory attribute.

target-ip: A list of addresses or prefixes under attack. This is an optional attribute.

target-port: A list of ports under attack. This is an optional attribute.

target-protocol: A list of protocols under attack. Valid protocol values include tcp, udp, sctp and dccp. This is an optional attribute.

Note: administrative-related clauses may be included as part of the

request (such a contract Identifier or a customer identifier). Those clauses are out of scope of this document.

The relative order of two rules is determined by comparing their respective policy identifiers. The rule with lower numeric policy identifier value has higher precedence (and thus will match before) than the rule with higher 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 and 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.

The following example shows POST request to signal that a Web-Service is under attack.

```
POST https://www.example.com/.well-known/v1/DOTS signal
Accept: application/json
Content-type: application/json
 {
   "policy-id": 123321333242,
   "target-ip": {"2002:db8:6401::1", "2002:db8:6401::1"},
   "target-port": {"80", "8080", "443"},
   "target-protocol": "tcp",
 }
```

Figure 3: POST to signal SOS

The DOTS server indicates the result of processing the POST request using HTTP response codes. HTTP 2xx codes are success whereas HTTP 4xx codes are some sort of invalid request. If the request is

missing one or more mandatory attributes then 400 (Bad Request) will be returned in the response or if the request contains invalid or unknown parameters then 400 (Invalid query) will be returned in the response. The HTTP response will include the JSON body received in the request.

5.1.2. Recall DOTS signal

An HTTP DELETE request will be used to delete an DOTS signal signaled to the DOTS server. If the DOTS server does not find the policy number conveyed in the DELETE request in its policy state data then it responds with 404 HTTP error response code.

```
DELETE {scheme}://{host}:{port}/.well-known/{URI suffix for DOTS signal}
Accept: application/json
Content-type: application/json
 {
   "policy-id": number
 }
```

Figure 4: Recall SOS

5.1.3. Retrieving DOTS signal

An HTTP GET request will be used to retrieve an DOTS signal signaled to the DOTS server. If the DOTS server does not find the policy number conveyed in the GET request in its policy state data then it responds with 404 HTTP error response code.

1) To retrieve all DOTS signal signaled by the DOTS client.

```
GET {scheme}://{host}:{port}/.well-known/{URI suffix for DOTS signal}
```

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.

```
GET {scheme}://{host}:{port}/.well-known/{URI suffix for DOTS signal}
Accept: application/json
Content-type: application/json
 {
   "policy-id": number
 }
```

Figure 5: GET to retrieve the rules

The following example shows the response of all the active policies on the DOTS server.


```
{
  "policy-data": [
      { "policy-id": 123321333242, "target-protocol": "tcp"},
      { "policy-id": 123321333244, "target-protocol": "udp"},
  ]
}
```

Figure 6: Response body

5.2. REST

A DOTS client could use HTTPS to provision and manage filters on the DOTS server. The DOTS client authenticates itself to the DOTS relay, which in turn authenticates itself to a DOTS server, creating a two-link chain of transitive authentication between the DOTS client and the DOTS server. The DOTS relay validates if the DOTS client is authorized to signal the black-list rules. Likewise, the DOTS server validates if the DOTS relay is authorized to signal the black-list rules. To create or purge filters, the DOTS client sends HTTP requests to the DOTS relay. The DOTS relay acts as an HTTP proxy, validates the rules and proxies the HTTP requests containing the black-listed IP addresses to the DOTS server. When the DOTS relay receives the associated HTTP response from the HTTP server, it propagates the response back to the DOTS client.

If an attack is detected by the DOTS relay then it can act as a HTTP client and signal the black-list rules to the DOTS server. Thus the DOTS relay plays the role of both HTTP client and HTTP proxy.

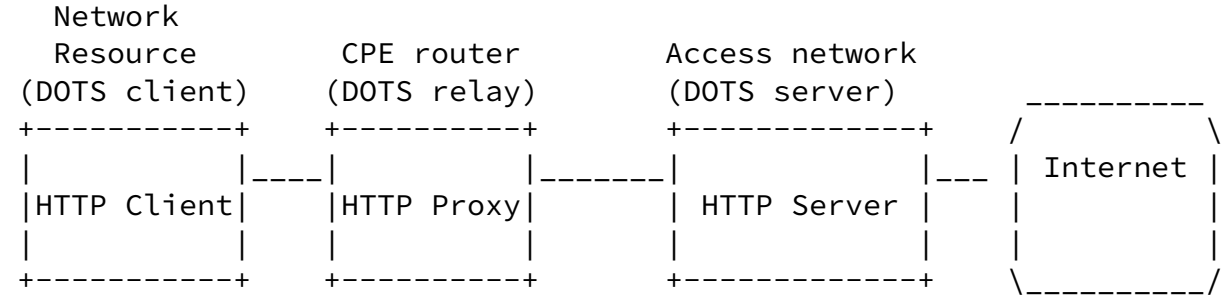

Figure 7

JSON [RFC7159] payloads can be used to convey both filtering rules as well as protocol-specific payload messages that convey request parameters and response information such as errors.

The figure above explains the protocol with a DOTS relay. The protocol is equally applicable to scenarios where a DOTS client directly talks to the DOTS server.

5.2.1. Filtering Rules

The following APIs define means for a DOTS client to configure filtering rules on a DOTS server.

5.2.1.1. Install filtering rules

An HTTP POST request will be used to push filtering rules to the DOTS server.

```
POST {scheme}://{host}:{port}/.well-known/{version}/{URI suffix for filtering
Accept: application/json
Content-type: application/json
{
   "policy-id": number,
   "traffic-protocol": string,
   "source-protocol-port": string,
   "destination-protocol-port": string,
   "destination-ip": string,
   "source-ip": string,
   "lifetime": number,
   "traffic-rate" : number,
 }
```

Figure 8: POST to install filtering rules

The header fields are described below.

policy-id: Identifier of the policy represented using a number. 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 client and used as an opaque value by the server. This document does not make any assumption about how this identifier is generated. This is an mandatory attribute.

traffic-protocol: Valid protocol values include tcp, udp, sctp and dccp. This is an mandatory attribute.

source-protocol-port: For TCP or UDP or SCTP or DCCP: the source range of ports (e.g., 1024-65535). This is an optional attribute.

destination-protocol-port: For TCP or UDP or SCTP or DCCP: the destination range of ports (e.g., 443-443). This information is useful to avoid disturbing a group of customers when address sharing is in use [RFC6269]. This is an optional attribute.

destination-ip: The destination IP addresses or prefixes. This is an optional attribute.

source-ip: The source IP addresses or prefixes. This is an optional attribute.

lifetime: Lifetime of the policy in seconds. Indicates the validity of a rule. Upon the expiry of this lifetime, and if the request is not reiterated, the rule will be withdrawn at the upstream network. The request can be reiterated by sending the same message again. The server always indicates the actual lifetime in the response. A null value is not allowed. This is an mandatory attribute.

traffic-rate: This field carries the rate information in IEEE floating point [IEEE.754.1985] format, units being bytes per second. A traffic-rate of '0' should result on all traffic for the particular flow to be discarded. This is an mandatory attribute.

The relative order of two rules is determined by comparing their respective policy identifiers. The rule with lower numeric policy identifier value has higher precedence (and thus will match before) than the rule with higher numeric policy identifier value.

Note: administrative-related clauses may be included as part of the request (such a contract Identifier or a customer identifier). Those clauses are out of scope of this document.

The following example shows POST request to block traffic from attacker IPv6 prefix 2001:db8:abcd:3f01::/64 to network resource using IPv6 address 2002:db8:6401::1 to provide HTTPS web service.

```
POST https://www.example.com/.well-known/v1/filter
Accept: application/json
Content-type: application/json
 {
   "policy-id": 123321333242,
   "traffic-protocol": "tcp",
   "source-protocol-port": "1-65535",
   "destination-protocol-port": "443",
   "destination-ip": "2001:db8:abcd:3f01::/64",
   "source-ip": "2002:db8:6401::1",
   "lifetime": 1800,
   "traffic-rate": 0,
 }
```

Figure 9: POST to install black-list rules

5.2.1.2. Remove filtering rules

An HTTP DELETE request will be used to delete filtering rules programmed on the DOTS server.

```
DELETE {scheme}://{host}:{port}/.well-known/{URI suffix for filtering}
Accept: application/json
Content-type: application/json
 {
   "policy-id": number
 }
```

Figure 10: DELETE to remove the rules

5.2.1.3. Retrieving installed filtering rules

An HTTP GET request will be used to retrieve filtering rules programmed on the DOTS server.

1) To retrieve all the black-lists rules programmed by the DOTS client.

```
GET {scheme}://{host}:{port}/.well-known/{URI suffix for filtering}
```

2) To retrieve specific black-list rules programmed by the DOTS cient.

```
GET {scheme}://{host}:{port}/.well-known/{URI suffix for filtering}
Accept: application/json
Content-type: application/json
 {
   "policy-id": number
 }
```

Figure 11: GET to retrieve the rules

6. IANA Considerations

TODO

7. 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 ciphersuite 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, attacker will 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.

8. Acknowledgements

Thanks to C. Jacquenet, Roland Dobbins, Andrew Mortensen, Roman D. Danyliw for the discussion and comments.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <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>.

[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>.

9.2. Informative References

[I-D.ietf-dots-requirements]
Mortensen, A., Moskowitz, R., and T. Reddy, "DDoS Open Threat Signaling Requirements", draft-ietf-dots-requirements-00 (work in progress), October 2015.

[I-D.ietf-tls-cached-info]
Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", draft-ietf-tls-cached-info-22 (work in progress), January 2016.

[I-D.ietf-tls-falsestart]
Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", draft-ietf-tls-falsestart-01 (work in progress), November 2015.

[REPORT] "Worldwide Infrastructure Security Report", 2014, <http://pages.arbornetworks.com/rs/arbor/images/WISR2014.pdf>.

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

[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>.

[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>.

[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>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[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>.

[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>.

[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>.

Appendix A. BGP

BGP defines a mechanism as described in [RFC5575] that can be used to automate inter-domain coordination of traffic filtering, such as what is required in order to mitigate DDoS attacks. However, support for BGP in an access network does not guarantee that traffic filtering will always be honored. Since a DOTS client will not receive an acknowledgment for the filtering request, the DOTS client should monitor and apply similar rules in its own network in cases where the DOTS server is unable to enforce the filtering rules. In addition, enforcement of filtering rules of BGP on Internet routers are usually governed by the maximum number of data elements the routers can hold as well as the number of events they are able to process in a given unit of time.

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

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Prashanth Patil
Cisco Systems, Inc.

Email: praspati@cisco.com

Mike Geller
Cisco Systems, Inc.
3250
Florida 33309
USA

Email: mgeller@cisco.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 42837
United States

Email: rgm@htt-consult.com