DOTS
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2016

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

October 18, 2015

Co-operative DDoS Mitigation
draft-reddy-dots-transport-01

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 April 20, 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:

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

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

- 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 certian 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.mortensen-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 notifiy the DOTS server that it is under a potential attack and request that the DOTS server take precautionary measures to mitigate the attack.

4. 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.

4.1. SOS

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

TBD: SOS messages SHOULD be exchanged over DTLS over UDP.

4.1.1. Signal SOS

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

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

Figure 1: POST to signal SOS

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. 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.

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.

To avoid SOS message fragmentation and the consequently decreased probability of message delivery, DOTS agents MUST ensure that the DTLS record MUST fit within a single datagram. DOTS agents can

exploit the fact that the IP specification [RFC0791] specifies that IP packets up to 576 bytes should never need to be fragmented, thus sending a maximum of 500 bytes of SOS message 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/SOS
Accept: application/json
Content-type: application/json
 {
   "policy-id": 123321333242,
   "target-ip": "2002:db8:6401::1",
   "target-port": "443",
   "target-protocol": "tcp",
 }
```

Figure 2: POST to signal SOS

4.1.2. Recall SOS

An HTTP DELETE request will be used to delete an SOS signaled to the DOTS server.

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

Figure 3: Recall SOS

4.1.3. Retrieving SOS

An HTTP GET request will be used to retrieve an SOS signaled to the DOTS server.

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

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

2) To retrieve a specific SOS signaled by the DOTS client.

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

Figure 4: GET to retrieve the rules

4.2. REST

A DOTS client could use HTTP 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.

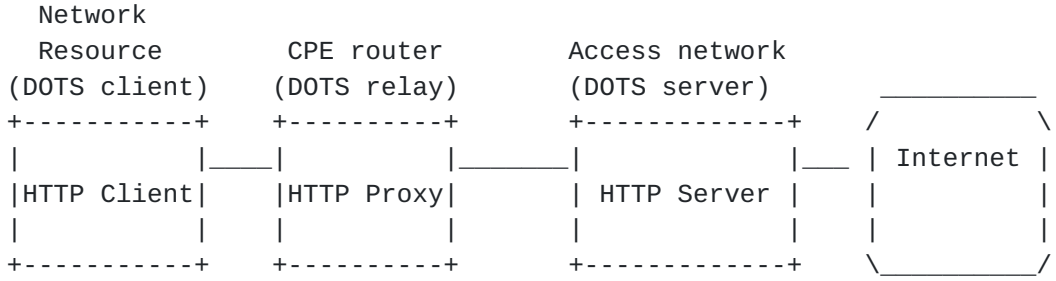

Figure 5

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.

4.2.1. Filtering Rules

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

4.2.1.1. 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 6: 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 same downstream network. 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.

traffic-protocol: Valid protocol values include tcp and udp.

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

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].

destination-ip: The destination IP addresses or prefixes.

source-ip: The source IP addresses or prefixes.

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. A null value is not allowed.

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.

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 7: POST to install black-list rules

4.2.1.2. 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 8: DELETE to remove the rules

4.2.1.3. 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 9: GET to retrieve the rules

5. IANA Considerations

TODO

6. Security Considerations

TODO

HTTPS MUST be used for data confidentiality and (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.

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.

7. Acknowledgements

Thanks to C. Jacquenet for the discussion and comments.

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>.

[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>.

8.2. Informative References

[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>.

[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>.

[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>.

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