BEHAVE
Internet-Draft
Expires: April 23, 2006

P. Srisuresh
Consultant
S. Sivakumar
K. Biswas
Cisco Systems
B. Ford
M.I.T
October 20, 2005

NAT Behavioral Requirements for TCP
draft-sivakumar-behave-nat-tcp-req-02

Abstract

NAT devices are available from a number of vendors and are in use by several residential and enterprise users. Yet, there is much

variation in how the NAT devices work. Application developers, network administrators and users of NAT devices seek some level of uniformity and predictability in how various of the NAT devices operate. The objective of this document is to specify the operational and behavioral requirements on the NAT devices while processing TCP packets. A NAT device that conforms to the requirements listed in the document will bring predictability in how NATs operate with regard to TCP packet processing. A NAT device is said to be IETF behave compliant when it complies with the requirements outlined in this document and other companion BEHAVE documents like [BEH-UDP], which outlines the requirements for processing UDP and some generic requirements.

Table of Contents

1. Introduction & Scope

NAT implementations vary amongst vendors in how they handle TCP packets. This document defines the operational and behavioral requirements that the NAT devices should comply with while processing TCP packets. In addition, a section is devoted to describing hints to implementers in deciphering some of the requirements.

The requirements outlined here are applicable across all NAT types identified in [RFC2663], most importantly the Traditional NAT, as described in [RFC3022]. This document does not mandate a specific implementation choice. Behavioral requirements for UDP are covered in [BEH-UDP].

Application Layer Gateways (ALGs) are out of scope for this document. However, hints on how a NAT could be extended to support ALGs are discussed under the hints section. Permissible ALGs are listed in [BEH-UDP] .

2. Terminology

Definitions for the NAT terms used throughout the document may be found in [RFC2663]. TCP terms used in the document are as per the definitions given in [TCP].

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. TCP requirements

This section lists the behavioral requirements of a NAT device when processing TCP packets. Associated with each requirement, the rationale behind the requirement is discussed in detail.

3.1 Address Mapping and/or TCP Port Mapping

NAT provides transparent routing between address realms by assigning realm-specific endpoint locator(s), as packets pertaining to a session cross realm boundaries. Several applications use the same endpoint to establish multiple simultaneous sessions. Many peer-to-peer applications use the public endpoint registration of peering hosts to initiate sessions into.

In order to support peer-to-peer applications and applications that entertain multiple simultaneous sessions using the same TCP endpoint, NAT MUST retain the association it assigned to an endpoint at the start of the first session and reuse the same endpoint association

when multiple sessions using the same endpoint are routed through the NAT device. Such a mapping between endpoints can occur when a NAT device maintains Address Mappings and/or TCP Port Mappings.

REQ-1: A NAT device MUST maintain Address and/or TCP Port Mappings

3.2 Timeouts for TCP NAT Sessions

As may be noted from [TCP], an end-to-end TCP session in its lifetime goes through three phases, namely Connecting, Established, and Closing. Each end-to-end TCP session is managed through a separate NAT Session within NAT. The NAT Session must be capable of identifying the current phase of the end-to-end TCP session it represents and use an idle timeout period that is appropriate for the current phase.

Connecting Phase: An end-to-end TCP session is said to enter the Connecting Phase when either of the endpoints sends the first SYN for the TCP session and exit the phase upon completion of 3-way SYN handshake. The idle timeout used by the NAT Session during this phase is called the SYN timeout. SYN timeout needs to be relatively short, so NAT can protect itself (and, potentially, the hosts behind it) from SYN flood attacks. A NAT session is freed when the SYN timeout expires

Established Phase: An end-to-end TCP session is said to enter the Established Phase upon completion of 3-way handshake and exit the phase upon seeing the first FIN or RST for the session. The idle timeout used by the NAT during this phase of the end-to-end TCP session is called the Session timeout. Session timeout needs to be relatively long, so the NAT Session can retain state of the end-to-end TCP Session within itself even after long periods of inactivity in the session. Long periods of inactivity is not uncommon with applications such as telnet and ftp. When Session timer expires, the corresponding NAT Session may be freed (or) the NAT Session may assume the TCP connection to have transitioned into the Closing phase.

Closing Phase: An end-to-end TCP session is said to enter the Closing Phase when either of the endpoitns sends the first FIN or RST for the session. Alternately, the NAT Session may deem the TCP session to have entered this phase when the TCP Session timer expires. The idle timeout used by the NAT Session during this phase is called the Close timeout. Close timeout is relatively short to ensure that the ACKs for the final FINs on a gracefully-closed TCP session had a chance to propagate in both directions, and to allow time for either endpoint to re-open a recently closed or reset TCP session if desired. A NAT device MAY opt to have different Close timeouts depending upon

whether the Closing phase is triggered by FIN or not. Once the Close timer expires, the NAT Session will be freed.

The following requirements apply to the NAT's timeouts:

REQ-2: A NAT device MUST be capable of identifying the current phase of an end-to-end TCP session and use different idle timeout periods for each phase of the TCP Session. The timeouts used for each phase SHOULD be admin configurable. The recommended value for SYN timeout is 30 seconds. The recommended value for TCP session timeout is 30 minutes. Lastly, the recommended value for close timeout is 2 x MSL (Maximum Segment Lifetime) or 4 minutes.

3.3 Handling Inbound SYN packets for an exisitng NAT TCP Session

A NAT device might allow sessions to be initiated in just one direction and not the other. However, once a NAT session is created for a permitted TCP session, and the TCP session is in Connecting phase, the NAT device MUST let the SYN packets through in either direction. This is because TCP protocol fundamentally permits simultaneous TCP Open from either end. A number of TCP based Peer-to-peer applications utilize the simultaneous TCP open technique to establish peer-to-peer connections.

If the TCP session is in established or Closing phases and a new SYN packet arrives, the NAT device should assume that the TCP session has re-entered the Connecting phase and initiate SYN timer. In summary, a NAT device MUST let the SYN packets through once a NAT Session is established for the TCP connection.

The following requirement applies to SYN packets arriving during Connecting, Established or Closing phases of a TCP connection.

REQ-3: A NAT device MUST let the SYN packets through when the SYN Packets are received on a TCP connection which is in one of Connecting, Established or Closing phases.

3.4 Handling Inbound SYN packets for non-existent TCP connections

Inbound SYN packets MUST be permitted on all NAT devices so long as an outbound SYN packet has been initiated first on the same connection tuple via the NAT device (and the NAT device has a NAT Session created for it). Inbound SYN packets are permitted, even without an outbound SYN on bi-directional NAT and load share NAT devices. In addition, NAT devices configured with a static inbound/birectional address or port mapping that matches the connection tuple of the inbound SYN packets MUST also permit the SYN packets.

In all other cases, the NAT devices MUST simply drop the inbound SYN packets. The NAT device should not send RST packet or an ICMP error packet back to the sender. However, there is an exception to this behavior. A NAT device sharing the TCP ports of the external IP address between NAT function and its own endhost applications may choose to respond with a RST when an inbound SYN is directed to one of the endhost TCP ports. However, when the inbound SYN is directed to one of the NAT function TCP ports, the response should be as described earlier for a NAT device.

The following requirement applies to incoming SYN packets.

REQ-4: A NAT device MUST let the incoming SYN packets through so long as an outbound SYN packet has been initiated first on the same connection tuple via the NAT device. NATs MUST permit inbound SYN even without an outbound SYN on bi-directional NAT & load share NAT devices. NATs MUST NOT generate RST or ICMP error packet back to the sender upon inbound SYN processing.

3.5 Denial of Service (DoS) attacks

Since NAT devices are Internet hosts, they can be the target of a number of different DoS attacks, such as SYN floods and RST attacks. NAT devices SHOULD employ the same sort of protection techniques as Internet-based servers do.

Let us examine two types of Dos attacks that are well known with regard to TCP connections. A SYN flood attack is a DoS attack in which one or more external entities initiate a number of simultaneous TCP connections using a SYN packet, but donot complete the 3-way handshake. Naturally, a NAT device is prone to this type of attack when the NAT device is in the traversal path of the SYN attacks. One technique to defend against this type of attack is to ensure that the NAT device employs a short SYN timeout and reduce the timeout even further when it determines it is under SYN flood attack.

RST attack is another well known DoS attack. An attacker could simply forge a number of RST packets for a variety of Established TCP connections and cause the NAT sessions to be reset and freed. One technique to defend against this type of attack is to validate the RST packet and not let the packet through unless the sequence number used in the RST packet is within the expected TCP window size of the TCP Session.

REQ-5: A NAT device SHOULD employ necessary techniques to defend against well known DoS attacks the endhosts are subject to.

3.6 NAT initiated TCP keep-alives

When session timer expires for a NAT session, that indicates that the associated TCP session has been idle with no activity for the period matching the TCP Session timeout. Sensing no activity, NAT could free up the NAT Session and remove the state associated with the TCP connection within the NAT device. However, doing so would violate the end-to-end reliability of the IP network. Ideally speaking, IP network is not supposed to retain any hard state. Unfortunately, a NAT device retains session state within itself (via the NAT Sessions) and this information should not be dropped without confirming that one or both halfs of the TCP session are alive.

A NAT device may validate the liveness of a TCP client by sending keep-alive packets to the TCP client using the technique described in section 4.2.3.6 of [HOST]. If the NAT device receives an ACK or other traffic from the internal endpoint, it resets the session timer and assumes the connection to be in ôEstablished" phase. If the NAT device receives a RST from the TCP client, the NAT device transitions the TCP connection into the "closing" phase and initiates Close timeout for the session. If the NAT device receives no response from the internal endpoint after sending several keep-alive packets, the NAT assumes that the internal endpoint is dead and again assumes that the TCP connection has entered the "closing" phase.

Below is the keep-alive requirement on NAT devices

REQ-6: When session timer expires for an established TCP connection, the NAT device MAY initiate sending TCP keep-alives to the clients prior to freeing up the Session state within NAT.

3.7 NAT initiated RST packets

When session timer expires for a NAT session, it is an indication that the associated TCP connection has been idle with no activity for the duration matching the TCP Session timeout. Sensing no activity, NAT could free up the NAT Session and remove the state associated with the TCP connection. When this happens, the two TCP endpoints in the network, which might potentially be alive, may be unable to resume activity on the connection because the NAT device enroute no longer has the state information pertaining to the end-to-end TCP connection. This can be problematic for application servers that impose limits on the number of connections a user might be allowed to setup in a given period of time. After a few zombie sessions, the server might deny access to its clients, when the connection count on the server exceeds the set limit. The Server has no way to know that some of the client sessions it retains are zombies and shouldnt be counted as real. In such a situation, the NAT device sending a RST

packet to both parties will alert the two parties of the connection going away. And, the application servers are not fooled with zombie sessions. Note, a NAT device may choose to send RST packets after it probed the TCP client with TCP Keep-alive packets.

Below is the requirement on sending TCP RST packet

REQ-7: When Session timer expires on an Established TCP connection, the NAT device SHOULD send a RST packet to both halfs of a TCP connection and enter Closing state on the connection prior to freeing up the NAT Session.

4. Hints to implementers

4.1 Light weight TCP state machine is a common practice

Unlike UDP, TCP sessions are fundamentally unicast in nature and multiple NAT Sessions cannot be aggregated. NAT devices maintain a separate NAT Session to track each end-to-end TCP connection that traverses the NAT device. A NAT device needs to be able to track the current phase of a TCP session at any given time so an idle timer for a duration appropriate for the phase is initiated. Further, a NAT device defending against even the most trivial type of DoS attack will require the knowledge of TCP sequence number and window Size to defend itself against such attacks. As such, many vendors use a light-weight state machine within the NAT Session to track the current state of a TCP connection. Items tracked within the state machine would include the last acknowledged sequence number from either half of the TCP session, TCP window size, and the TCP connection phase.

The State machine within a NAT Session enters the Connecting state when NAT sees the first SYN packet for that session. The state machine transitions from the Connecting to Established state once the 3-way handshake is completed. The state machine transitions from the Established state to the Closing state when the NAT observes a FIN/FIN ACK sequence, representing graceful shutdown reached cooperatively by both endpoints, or when the NAT observes a RST from either endpoint, representing a non-graceful connection reset forced by one endpoint. The NAT device deletes the NAT session after the Close timer expires while the TCP connection is in the Closing state.

In addition to the basic state information, NAT devices also record information about TCP sequence numbers and acknowledgment numbers on the traversing TCP packets, so as to defend against DOS attacks (REQ-5). If the NAT contains built-in ALGs that can change the payload length of TCP packets, effectively inserting or removing bytes from the TCP stream in one or both directions, then the NAT

MUST adjust the sequence numbers in all subsequent packets exchanged in either direction to reflect these inserted or removed bytes.

4.2 TCP segment processing in NATs supporting ALGs

The following discussion on TCP segment processing is relevant only when a NAT device includes support for one or more embedded ALGs. Many NAT devices have the ALG for FTP enabled by default. A NAT device may receive payload relevant to an ALG in multiple TCP segments. Consider the following diagram where the MSS is set to 536 bytes in each endpoint of the TCP connection.

```
   +--------------------+                         +--------------------+
   | Application-Layer |                         | Application-Layer |
   +--------------------+                         +--------------------+
   | TCP [MSS = 536]   |                         | TCP [MSS = 536]   |
   +--------------------+                         +--------------------+
   | IP                |                         | IP                |
   +--------------------+                         +--------------------+
   | Lower-Layer       |                         | Lower-Layer       |
   |                   |                         |                   |
   +--------------------+                         +--------------------+
        End-host -1                                    End-host -2
            |                   +---------+                 |
            +-------------------| NAT     |-----------------+
                                +---------+
```

Say the application layer on end-host-1 is sending a payload of size 600 bytes. The payload will be sent to the recipient in 2 TCP segments as follows, because the MSS is set to 536 bytes.

```
       TCP Segment 1:
       +---------+-----------------------------------------+----------+
       |IP header| TCP header[Payload-Len=536,PUSH flag = 0]|Appl-data1|
       +---------+-----------------------------------------+----------+

       TCP Segment 2:
       +---------+----------------------------------------+----------+
       |IP header| TCP header[Payload-Len=64,PUSH flag = 1]|Appl-data2|
       +---------+----------------------------------------+----------+
```

A NAT device enroute may receive the TCP segments either in order or out of order. In either case, the NAT device needs to assemble the individual segments into a contiguous payload and make the complete payload available for the ALG to process prior to forwarding the

segments transparently to another realm.

In order to do this, a NAT device is required to enforce some type of queuing mechanism such that when all relevant segments of a payload are received, it is able to reassemble the TCP segments and make the contiguous payload available for ALG processing.

For in-order segments, the NAT device needs to send a TCP ACK for the initial segments it received, but didnt forward to the recipient enpoint. This is done so the NAT device can prompt the sending endpoint to continue to send the remaining TCP segments.

4.3 Adjusting Sequence Acknowledgement Numbers

The following discussion on adjusting Sequence and Acknowledgement numbers is relevant only when a NAT device includes support for one or more embedded ALGs.

When the embedded ALG on a NAT device modifies the TCP payload, the corresponding payload may increase or decrease in size. As a result, the NAT device is expected to remember the delta change and adjust sequence/acknowledgement numbers in all subsequent TCP packets within the session. Implementors of NAT devices often keep the delta changes in payload due to ALG processing within the NAT Session as an extension of the state information the NAT device keeps.

5. TCP behavioral requirements summary

Below is a summary of all the TCP behavioral requirements.

REQ-1: A NAT device MUST maintain Address and/or TCP Port Mappings.

REQ-2: A NAT device MUST be capable of identifying the current phase of an end-to-end TCP session and use different idle timeout periods for each phase of the TCP Session.

The timeouts used for each phase SHOULD be admin configurable. The recommended value for SYN timeout is 30 seconds. The recommended value for TCP session timeout is 30 minutes. Lastly, the recommended value for close timeout is 2 x MSL (Maximum Segment Lifetime) or 4 minutes.

REQ-3: A NAT device MUST let the SYN packets through when the SYN Packets are received on a TCP connection which is in one of Connecting or Closing phase.

REQ-4: A NAT device MUST let the incoming SYN packets through so long as an outbound SYN packet has been initiated first on the same

connection tuple via the NAT device. NATs MUST permit inbound SYN even without an outbound SYN on bi-directional NAT & load share NAT devices. NATs MUST NOT generate RST or ICMP error packet back to the sender upon inbound SYN processing.

REQ-5: A NAT device SHOULD employ necessary techniques to defend against well known DoS attacks the endhosts are subject to.

REQ-6: When session timer expires for an Established TCP connection, the NAT device MAY initiate sending TCP keep-alives to the clients prior to freeing up the Session state within NAT.

REQ-7: When session timer expires for an Established TCP connection, the NAT device SHOULD send a RST packet to both halfs of a TCP connection and enter Closing state on the connection prior to freeing Up the NAT Session.

6. Security considerations

The security considerations described in [RFC2663] for all variations of NATs are applicable here. The recommendations and requirements in this document do not effect the security properties of the NAT devices adversely.

7. Acknowledgements

The authors would like to thank Nagendra Modadugu, Saikat Guha and other BEHAVE workgroup members for their valuable feedback and comments.

8. References

8.1 Normative References

[HOST]   Braden, R., "Requirements for Internet-Hosts - Communication Layers", RFC 1122, October 1998.

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

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[TCP]   Postel, J., "Transmission Control Protocol (TCP) Specification", RFC 793, STD 7, September 1981.

8.2 Informative References

[ASND]   Reynolds, J. and J. Postel, "Assigned numbers", RFC 923, October 1984.

[BEH-UDP] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP, draft-ietf-behave-nat-00.txt (work-in-progress)", January 2005.

[ICMP]   Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[NAT-CHK] Ford, B. and D. Andersen, "Nat Check Web Site: http://midcom-p2p.sourceforge.net", June 2004.

[NAT-CMPL] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, January 2001.

[UNSAF]   Daigle, L. and IAB, "IAB Considerations for Unilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[V4-REQ]   Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

Authors' Addresses

Pyda Srisuresh
Consultant
20072 Pacifica Dr.
Cupertino, CA 95014
USA

Phone: (408)836-4773
Email: srisuresh@yahoo.com

Senthil Sivakumar
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
USA

Phone:
Email: ssenthil@cisco.com

Kaushik Biswas
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
USA

Phone: +1 408 525 5134
Email: kbiswas@cisco.com

Bryan Ford
M.I.T.
Laboratory for Computer Science
77 Massachusetts Ave.
Cambridge, MA 02139
USA

Phone: 1-617-253-5261
Email: baford@mit.edu

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.