Network Working Group
Request for Comments: 5382
BCP: 142
Category: Best Current Practice

S. Guha, Ed.
Cornell U.
K. Biswas
Cisco Systems
B. Ford
MPI-SWS
S. Sivakumar
Cisco Systems
P. Srisuresh
Kazeon Systems

October 2008

NAT Behavioral Requirements for TCP

Status of This Memo

Abstract

This document defines a set of requirements for NATs that handle TCP that would allow many applications, such as peer-to-peer applications and online games to work consistently. Developing NATs that meet this set of requirements will greatly increase the likelihood that these applications will function properly.

Table of Contents

1. Applicability Statement

This document is adjunct to [BEHAVE-UDP], which defines many terms relating to NATs, lays out general requirements for all NATs, and sets requirements for NATs that handle IP and unicast UDP traffic. The purpose of this document is to set requirements for NATs that handle TCP traffic.

The requirements of this specification apply to traditional NATs as described in [RFC2663].

This document only covers the TCP aspects of NAT traversal. Middlebox behavior that is not necessary for network address translation of TCP is out of scope. Packet inspection above the TCP layer and firewalls are out of scope except for Application Level Gateway (ALG) behavior that may interfere with NAT traversal. Application and OS aspects of TCP NAT traversal are out of scope. Signaling-based approaches to NAT traversal, such as Middlebox Communication (MIDCOM) and Universal Plug and Play (UPnP), that directly control the NAT are out of scope. Finally, TCP connections intended for the NAT (e.g., an HTTP or Secure Shell Protocol (SSH) management interface) and TCP connections initiated by the NAT (e.g., reliable syslog client) are out of scope.

2. Introduction

Network Address Translators (NATs) hinder connectivity in applications where sessions may be initiated to internal hosts. Readers may refer to [RFC3022] for detailed information on traditional NATs. [BEHAVE-UDP] lays out the terminology and requirements for NATs in the context of IP and UDP. This document supplements these by setting requirements for NATs that handle TCP traffic. All definitions and requirements in [BEHAVE-UDP] are inherited here.

[RFC4614] chronicles the evolution of TCP from the original definition [RFC0793] to present-day implementations. While much has changed in TCP with regards to congestion control and flow control, security, and support for high-bandwidth networks, the process of initiating a connection (i.e., the 3-way handshake or simultaneous-open) has changed little. It is the process of connection initiation that NATs affect the most. Experimental approaches such as T/TCP [RFC1644] have proposed alternate connection initiation approaches, but have been found to be complex and susceptible to denial-of-service attacks. Modern operating systems and NATs consequently primarily support the 3-way handshake and simultaneous-open modes of connection initiation as described in [RFC0793].

Recently, many techniques have been devised to make peer-to-peer TCP applications work across NATs. [STUNT], [NATBLASTER], and [P2PNAT] describe Unilateral Self-Address Fixing (UNSAF) mechanisms that allow peer-to-peer applications to establish TCP through NATs. These approaches require only endpoint applications to be modified and work with standards compliant OS stacks. The approaches, however, depend on specific NAT behavior that is usually, but not always, supported by NATs (see [TCPTRAV] and [P2PNAT] for details). Consequently, a complete TCP NAT traversal solution is sometimes forced to rely on public TCP relays to traverse NATs that do not cooperate. This document defines requirements that ensure that TCP NAT traversal approaches are not forced to use data relays.

3. 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].

"NAT" in this specification includes both "Basic NAT" and "Network Address/Port Translator (NAPT)" [RFC2663]. The term "NAT Session" is adapted from [NAT-MIB] and is defined as follows.

NAT Session - A NAT session is an association between a TCP session as seen in the internal realm and a TCP session as seen in the external realm, by virtue of NAT translation. The NAT session will provide the translation glue between the two session representations.

This document uses the term "TCP connection" (or just "connection") to refer to individual TCP flows identified by the 4-tuple (source and destination IP address and TCP port) and the initial sequence numbers (ISN).

This document uses the term "address and port mapping" (or just "mapping") as defined in [BEHAVE-UDP] to refer to state at the NAT necessary for network address and port translation of TCP connections. This document also uses the terms "Endpoint-Independent Mapping", "Address-Dependent Mapping", "Address and Port-Dependent Mapping", "filtering behavior", "Endpoint-Independent Filtering", "Address-Dependent Filtering", "Address and Port-Dependent Filtering", "Port assignment", "Port overloading", "hairpinning", and "External source IP address and port" as defined in [BEHAVE-UDP].

4. TCP Connection Initiation

This section describes various NAT behaviors applicable to TCP connection initiation.

4.1. Address and Port Mapping Behavior

A NAT uses a mapping to translate packets for each TCP connection. A mapping is dynamically allocated for connections initiated from the internal side, and potentially reused for certain subsequent connections. NAT behavior regarding when a mapping can be reused differs for different NATs as described in [BEHAVE-UDP].

Consider an internal IP address and TCP port (X:x) that initiates a TCP connection to an external (Y1:y1) tuple. Let the mapping allocated by the NAT for this connection be (X1':x1'). Shortly thereafter, the endpoint initiates a connection from the same (X:x) to an external address (Y2:y2) and gets the mapping (X2':x2') on the NAT. As per [BEHAVE-UDP], if (X1':x1') equals (X2':x2') for all values of (Y2:y2), then the NAT is defined to have "Endpoint-Independent Mapping" behavior. If (X1':x1') equals (X2':x2') only when Y2 equals Y1, then the NAT is defined to have "Address-Dependent Mapping" behavior. If (X1':x1') equals (X2':x2') only when (Y2:y2) equals (Y1:y1), possible only for consecutive connections to the same external address shortly after the first is terminated and if the NAT retains state for connections in TIME_WAIT state, then the NAT is defined to have "Address and Port-Dependent Mapping" behavior. This document introduces one additional behavior where (X1':x1') never equals (X2':x2'), that is, for each connection a new mapping is allocated; in such a case, the NAT is defined to have "Connection-Dependent Mapping" behavior.

REQ-1: A NAT MUST have an "Endpoint-Independent Mapping" behavior for TCP.

Justification: REQ-1 is necessary for UNSAF methods to work. Endpoint-Independent Mapping behavior allows peer-to-peer applications to learn and advertise the external IP address and port allocated to an internal endpoint such that external peers can contact it (subject to the NAT's security policy). The security policy of a NAT is independent of its mapping behavior and is discussed later in Section 4.3. Having Endpoint-Independent Mapping behavior allows peer-to-peer applications to work consistently without compromising the security benefits of the NAT.

4.2. Internally Initiated Connections

An internal endpoint initiates a TCP connection through a NAT by sending a SYN packet. The NAT allocates (or reuses) a mapping for the connection, as described in the previous section. The mapping defines the external IP address and port used for translation of all packets for that connection. In particular, for client-server

applications where an internal client initiates the connection to an external server, the mapping is used to translate the outbound SYN, the resulting inbound SYN-ACK response, the subsequent outbound ACK, and other packets for the connection. This method of connection initiation corresponds to the 3-way handshake (defined in [RFC0793]) and is supported by all NATs.

Peer-to-peer applications use an alternate method of connection initiation termed simultaneous-open (Fig. 8, [RFC0793]) to traverse NATs. In the simultaneous-open mode of operation, both peers send SYN packets for the same TCP connection. The SYN packets cross in the network. Upon receiving the other end's SYN packet, each end responds with a SYN-ACK packet, which also cross in the network. The connection is considered established once the SYN-ACKs are received. From the perspective of the NAT, the internal host's SYN packet is met by an inbound SYN packet for the same connection (as opposed to a SYN-ACK packet during a 3-way handshake). Subsequent to this exchange, both an outbound and an inbound SYN-ACK are seen for the connection. Some NATs erroneously block the inbound SYN for the connection in progress. Some NATs block or incorrectly translate the outbound SYN-ACK. Such behavior breaks TCP simultaneous-open and prevents peer-to-peer applications from functioning correctly behind a NAT.

In order to provide network address translation service for TCP, it is necessary for a NAT to correctly receive, translate, and forward all packets for a connection that conform to valid transitions of the TCP State-Machine (Fig. 6, [RFC0793]).

REQ-2: A NAT MUST support all valid sequences of TCP packets (defined in [RFC0793]) for connections initiated both internally as well as externally when the connection is permitted by the NAT. In particular:

a) In addition to handling the TCP 3-way handshake mode of connection initiation, A NAT MUST handle the TCP simultaneous-open mode of connection initiation.

Justification: The intent of this requirement is to allow standards compliant TCP stacks to traverse NATs no matter what path the stacks take through the TCP state-machine and no matter which end initiates the connection as long as the connection is permitted by the filtering policy of the NAT (filtering policy is described in the following section).

a) In addition to TCP packets for a 3-way handshake, A NAT must be prepared to accept an inbound SYN and an outbound SYN-ACK for an internally initiated connection in order to support simultaneous-open.

4.3. Externally Initiated Connections

The NAT allocates a mapping for the first connection initiated by an internal endpoint to an external endpoint. In some scenarios, the NAT's policy may allow this mapping to be reused for connections initiated from the external side to the internal endpoint. Consider as before an internal IP address and port (X:x) that is assigned (or reuses) a mapping (X1':x1') when it initiates a connection to an external (Y1:y1). An external endpoint (Y2:y2) attempts to initiate a connection with the internal endpoint by sending a SYN to (X1':x1'). A NAT can choose to either allow the connection to be established, or to disallow the connection. If the NAT chooses to allow the connection, it translates the inbound SYN and routes it to (X:x) as per the existing mapping. It also translates the SYN-ACK generated by (X:x) in response and routes it to (Y2:y2), and so on. Alternately, the NAT can disallow the connection by filtering the inbound SYN.

A NAT may allow an existing mapping to be reused by an externally initiated connection if its security policy permits. Several different policies are possible as described in [BEHAVE-UDP]. If a NAT allows the connection initiation from all (Y2:y2), then it is defined to have "Endpoint-Independent Filtering" behavior. If the NAT allows connection initiations only when Y2 equals Y1, then the NAT is defined to have "Address-Dependent Filtering" behavior. If the NAT allows connection initiations only when (Y2:y2) equals (Y1:y1), then the NAT is defined to have "Address and Port-Dependent Filtering" behavior (possible only shortly after the first connection has been terminated but the mapping is still active). One additional filtering behavior defined in this document is when the NAT does not allow any connection initiations from the external side; in such cases, the NAT is defined to have "Connection-Dependent Filtering" behavior. The difference between "Address and Port-Dependent Filtering" and "Connection-Dependent Filtering" behavior is that the former permits an inbound SYN during the TIME_WAIT state of the first connection to initiate a new connection while the latter does not.

REQ-3: If application transparency is most important, it is RECOMMENDED that a NAT have an "Endpoint-Independent Filtering" behavior for TCP. If a more stringent filtering behavior is most important, it is RECOMMENDED that a NAT have an "Address-Dependent Filtering" behavior.

a) The filtering behavior MAY be an option configurable by the administrator of the NAT.
b) The filtering behavior for TCP MAY be independent of the filtering behavior for UDP.

Justification: The intent of this requirement is to allow peer-to-peer applications that do not always initiate connections from the internal side of the NAT to continue to work in the presence of NATs. This behavior also allows applications behind a BEHAVE compliant NAT to inter-operate with remote endpoints that are behind non-BEHAVE compliant (legacy) NATs. If the remote endpoint's NAT does not have Endpoint-Independent Mapping behavior but has only one external IP address, then an application can still traverse the combination of the two NATs if the local NAT has Address-Dependent Filtering. Section 9 contains a detailed discussion on the security implications of this requirement.

If the inbound SYN packet is filtered, either because a corresponding mapping does not exist or because of the NAT's filtering behavior, a NAT has two basic choices: to ignore the packet silently, or to signal an error to the sender. Signaling an error through ICMP messages allows the sender to quickly detect that the SYN did not reach the intended destination. Silently dropping the packet, on the other hand, allows applications to perform simultaneous-open more reliably.

Silently dropping the SYN aids simultaneous-open as follows. Consider that the application is attempting a simultaneous-open and the outbound SYN from the internal endpoint has not yet crossed the NAT (due to network congestion or clock skew between the two endpoints); this outbound SYN would otherwise have created the necessary mapping at the NAT to allow translation of the inbound SYN. Since the outbound SYN did not reach the NAT in time, the inbound SYN cannot be processed. If a NAT responds to the premature inbound SYN with an error message that forces the external endpoint to abandon the connection attempt, it hinders applications performing a TCP simultaneous-open. If instead the NAT silently ignores the inbound SYN, the external endpoint retransmits the SYN after a TCP timeout. In the meantime, the NAT creates the mapping in response to the (delayed) outbound SYN such that the retransmitted inbound SYN can be routed and simultaneous-open can succeed. The downside to this behavior is that in the event the inbound SYN is erroneous, the remote side does not learn of the error until after several TCP timeouts.

NAT support for simultaneous-open as well as quickly signaling errors are both important for applications. Unfortunately, there is no way for a NAT to signal an error without forcing the endpoint to abort a potential simultaneous-open: TCP RST and ICMP Port Unreachable packets require the endpoint to abort the attempt while the ICMP Host and Network Unreachable errors may adversely affect other connections to the same host or network [RFC1122].

In addition, when an unsolicited SYN is received by the NAT, the NAT may not know whether the application is attempting a simultaneous-open (and that it should therefore silently drop the SYN) or whether the SYN is in error (and that it should notify the sender).

REQ-4: A NAT MUST NOT respond to an unsolicited inbound SYN packet for at least 6 seconds after the packet is received. If during this interval the NAT receives and translates an outbound SYN for the connection the NAT MUST silently drop the original unsolicited inbound SYN packet. Otherwise, the NAT SHOULD send an ICMP Port Unreachable error (Type 3, Code 3) for the original SYN, unless REQ-4a applies.

a) The NAT MUST silently drop the original SYN packet if sending a response violates the security policy of the NAT.

Justification: The intent of this requirement is to allow simultaneous-open to work reliably in the presence of NATs as well as to quickly signal an error in case the unsolicited SYN is in error. As of writing this memo, it is not possible to achieve both; the requirement therefore represents a compromise. The NAT should tolerate some delay in the outbound SYN for a TCP simultaneous-open, which may be due to network congestion or loose synchronization between the endpoints. If the unsolicited SYN is not part of a simultaneous-open attempt and is in error, the NAT should endeavor to signal the error in accordance with [RFC1122].

a) There may, however, be reasons for the NAT to rate-limit or omit such error notifications, for example, in the case of an attack. Silently dropping the SYN packet when under attack allows simultaneous-open to work without consuming any extra network bandwidth or revealing the presence of the NAT to attackers. Section 9 mentions the security considerations for this requirement.

For NATs that combine NAT functionality with end-host functionality (e.g., an end-host that also serves as a NAT for other hosts behind it), REQ-4 above applies only to SYNs intended for the NAT'ed hosts and not to SYNs intended for the NAT itself. One way to determine whether the inbound SYN is intended for a NAT'ed host is to allocate NAT mappings from one port range, and allocate ports for local endpoints from a different non-overlapping port range. More dynamic implementations can be imagined.

5. NAT Session Refresh

A NAT maintains state associated with in-progress and established connections. Because of this, a NAT is susceptible to a resource-exhaustion attack whereby an attacker (or virus) on the internal side attempts to cause the NAT to create more state than for which it has resources. To prevent such an attack, a NAT needs to abandon sessions in order to free the state resources.

A common method that is applicable only to TCP is to preferentially abandon sessions for crashed endpoints, followed by closed TCP connections and partially open connections. A NAT can check if an endpoint for a session has crashed by sending a TCP keep-alive packet and receiving a TCP RST packet in response. If the NAT cannot determine whether the endpoint is active, it should not abandon the session until the TCP connection has been idle for some time. Note that an established TCP connection can stay idle (but live) indefinitely; hence, there is no fixed value for an idle-timeout that accommodates all applications. However, a large idle-timeout motivated by recommendations in [RFC1122] can reduce the chances of abandoning a live session.

A TCP connection passes through three phases: partially open, established, and closing. During the partially open phase, endpoints synchronize initial sequence numbers. The phase is initiated by the first SYN for the connection and extends until both endpoints have sent a packet with the ACK flag set (TCP states: SYN_SENT and SYN_RCVD). ACKs in both directions mark the beginning of the established phase where application data can be exchanged indefinitely (TCP states: ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, and CLOSE_WAIT). The closing phase begins when both endpoints have terminated their half of the connection by sending a FIN packet. Once FIN packets are seen in both directions, application data can no longer be exchanged, but the stacks still need to ensure that the FIN packets are received (TCP states: CLOSING and LAST_ACK).

TCP connections can stay in established phase indefinitely without exchanging any packets. Some end-hosts can be configured to send keep-alive packets on such idle connections; by default, such keep-alive packets are sent every 2 hours if enabled [RFC1122]. Consequently, a NAT that waits for slightly over 2 hours can detect idle connections with keep-alive packets being sent at the default rate. TCP connections in the partially open or closing phases, on the other hand, can stay idle for at most 4 minutes while waiting for in-flight packets to be delivered [RFC1122].

The "established connection idle-timeout" for a NAT is defined as the minimum time a TCP connection in the established phase must remain idle before the NAT considers the associated session a candidate for removal. The "transitory connection idle-timeout" for a NAT is defined as the minimum time a TCP connection in the partially open or closing phases must remain idle before the NAT considers the associated session a candidate for removal. TCP connections in the TIME_WAIT state are not affected by the "transitory connection idle-timeout".

REQ-5: If a NAT cannot determine whether the endpoints of a TCP connection are active, it MAY abandon the session if it has been idle for some time. In such cases, the value of the "established connection idle-timeout" MUST NOT be less than 2 hours 4 minutes. The value of the "transitory connection idle-timeout" MUST NOT be less than 4 minutes.
a) The value of the NAT idle-timeouts MAY be configurable.

Justification: The intent of this requirement is to minimize the cases where a NAT abandons session state for a live connection. While some NATs may choose to abandon sessions reactively in response to new connection initiations (allowing idle connections to stay up indefinitely in the absence of new initiations), other NATs may choose to proactively reap idle sessions. In cases where the NAT cannot actively determine if the connection is alive, this requirement ensures that applications can send keep-alive packets at the default rate (every 2 hours) such that the NAT can passively determine that the connection is alive. The additional 4 minutes allows time for in-flight packets to cross the NAT.

NAT behavior for handling RST packets, or connections in TIME_WAIT state is left unspecified. A NAT MAY hold state for a connection in TIME_WAIT state to accommodate retransmissions of the last ACK. However, since the TIME_WAIT state is commonly encountered by internal endpoints properly closing the TCP connection, holding state for a closed connection may limit the throughput of connections through a NAT with limited resources. [RFC1337] describes hazards associated with TIME_WAIT assassination.

The handling of non-SYN packets for connections for which there is no active mapping is left unspecified. Such packets may be received if the NAT silently abandons a live connection, or abandons a connection in TIME_WAIT state before the 4 minute TIME_WAIT period expires. The decision to either silently drop such packets or to respond with a TCP RST packet is left up to the implementation.

NAT behavior for notifying endpoints when abandoning live connections is left unspecified. When a NAT abandons a live connection, for example due to a timeout expiring, the NAT MAY either send TCP RST packets to the endpoints or MAY silently abandon the connection.

Sending a RST notification allows endpoint applications to recover more quickly; however, notifying the endpoints may not always be possible if, for example, session state is lost due to a power failure.

6. Application Level Gateways

Application Level Gateways (ALGs) in certain NATs modify IP addresses and TCP ports embedded inside application protocols. Such ALGs may interfere with UNSAF methods or protocols that try to be NAT-aware and must therefore be used with extreme caution.

REQ-6: If a NAT includes ALGs that affect TCP, it is RECOMMENDED that all of those ALGs (except for FTP [RFC0959]) be disabled by default.

Justification: The intent of this requirement is to prevent ALGs from interfering with UNSAF methods. The default state of an FTP ALG is left unspecified because of legacy concerns: as of writing this memo, a large fraction of legacy FTP clients do not enable passive (PASV) mode by default and require an ALG to traverse NATs.

7. Other Requirements Applicable to TCP

A list of general and UDP-specific NAT behavioral requirements are described in [BEHAVE-UDP]. A list of ICMP-specific NAT behavioral requirements are described in [BEHAVE-ICMP]. The requirements listed below reiterate the requirements from these two documents that directly affect TCP. The following requirements do not relax any requirements in [BEHAVE-UDP] or [BEHAVE-ICMP].

7.1. Port Assignment

NATs that allow different internal endpoints to simultaneously use the same mapping are defined in [BEHAVE-UDP] to have a "Port assignment" behavior of "Port overloading". Such behavior is undesirable, as it prevents two internal endpoints sharing the same mapping from establishing simultaneous connections to a common external endpoint.

REQ-7: A NAT MUST NOT have a "Port assignment" behavior of "Port overloading" for TCP.

Justification: This requirement allows two applications on the internal side of the NAT to consistently communicate with the same destination.

NAT behavior for preserving the source TCP port range for connections is left unspecified. Some applications expect the source TCP port to be in the well-known range (TCP ports from 0 to 1023). The "r" series of commands (rsh, rcp, rlogin, etc.) are an example. NATs that preserve the range from which the source port is picked allow such applications to function properly through the NAT; however, by doing so the NAT may compromise the security of the application in certain situations; applications that depend only on the IP address and source TCP port range for security (the "r" commands, for example) cannot distinguish between an attacker and a legitimate user behind the same NAT.

7.2. Hairpinning Behavior

NATs that forward packets originating from an internal address, destined for an external address that matches the active mapping for an internal address, back to that internal address are defined in [BEHAVE-UDP] as supporting "hairpinning". If the NAT presents the hairpinned packet with an external source IP address and port (i.e., the mapped source address and port of the originating internal endpoint), then it is defined to have "External source IP address and port" for hairpinning. Hairpinning is necessary to allow two internal endpoints (known to each other only by their external mapped addresses) to communicate with each other. "External source IP address and port" behavior for hairpinning avoids confusing implementations that expect the external source IP address and port.

REQ-8: A NAT MUST support "hairpinning" for TCP.
   a) A NAT's hairpinning behavior MUST be of type "External source IP address and port".

Justification: This requirement allows two applications behind the same NAT that are trying to communicate with each other using their external addresses.
   a) Using the external source address and port for the hairpinned packet is necessary for applications that do not expect to receive a packet from a different address than the external address they are trying to communicate with.

7.3. ICMP Responses to TCP Packets

Several TCP mechanisms depend on the reception of ICMP error messages triggered by the transmission of TCP segments. One such mechanism is path MTU discovery [RFC1191], which is required for the correct

operation of TCP. The current path MTU discovery mechanism requires the sender of TCP segments to be notified of ICMP "Datagram Too Big" responses.

REQ-9: If a NAT translates TCP, it SHOULD translate ICMP Destination Unreachable (Type 3) messages.

Justification: Translating ICMP Destination Unreachable messages, particularly the "Fragmentation Needed and Don't Fragment was Set" (Type 3, Code 4) message avoids communication failures ("black holes" [RFC2923]). Furthermore, TCP's connection establishment and maintenance mechanisms also behave much more efficiently when ICMP Destination Unreachable messages arrive in response to outgoing TCP segments.

REQ-10: Receipt of any sort of ICMP message MUST NOT terminate the NAT mapping or TCP connection for which the ICMP was generated.

Justification: This is necessary for reliably performing TCP simultaneous-open where a remote NAT may temporarily signal an ICMP error.

8. Requirements

A NAT that supports all of the mandatory requirements of this specification (i.e., the "MUST") and is compliant with [BEHAVE-UDP], is "compliant with this specification". A NAT that supports all of the requirements of this specification (i.e., included the "RECOMMENDED") and is fully compliant with [BEHAVE-UDP] is "fully compliant with all the mandatory and recommended requirements of this specification".

REQ-1: A NAT MUST have an "Endpoint-Independent Mapping" behavior for TCP.

REQ-2: A NAT MUST support all valid sequences of TCP packets (defined in [RFC0793]) for connections initiated both internally as well as externally when the connection is permitted by the NAT. In particular:

a) In addition to handling the TCP 3-way handshake mode of connection initiation, A NAT MUST handle the TCP simultaneous-open mode of connection initiation.

REQ-3: If application transparency is most important, it is RECOMMENDED that a NAT have an "Endpoint-Independent Filtering" behavior for TCP. If a more stringent filtering behavior is most important, it is RECOMMENDED that a NAT have an "Address-Dependent Filtering" behavior.

a) The filtering behavior MAY be an option configurable by the administrator of the NAT.
b) The filtering behavior for TCP MAY be independent of the filtering behavior for UDP.

REQ-4: A NAT MUST NOT respond to an unsolicited inbound SYN packet for at least 6 seconds after the packet is received. If during this interval the NAT receives and translates an outbound SYN for the connection the NAT MUST silently drop the original unsolicited inbound SYN packet. Otherwise, the NAT SHOULD send an ICMP Port Unreachable error (Type 3, Code 3) for the original SYN, unless REQ-4a applies.
a) The NAT MUST silently drop the original SYN packet if sending a response violates the security policy of the NAT.

REQ-5: If a NAT cannot determine whether the endpoints of a TCP connection are active, it MAY abandon the session if it has been idle for some time. In such cases, the value of the "established connection idle-timeout" MUST NOT be less than 2 hours 4 minutes. The value of the "transitory connection idle-timeout" MUST NOT be less than 4 minutes.
a) The value of the NAT idle-timeouts MAY be configurable.

REQ-6: If a NAT includes ALGs that affect TCP, it is RECOMMENDED that all of those ALGs (except for FTP [RFC0959]) be disabled by default.

The following requirements reiterate requirements from [BEHAVE-UDP] or [BEHAVE-ICMP] that directly affect TCP. This document does not relax any requirements in [BEHAVE-UDP] or [BEHAVE-ICMP].

REQ-7: A NAT MUST NOT have a "Port assignment" behavior of "Port overloading" for TCP.

REQ-8: A NAT MUST support "hairpinning" for TCP.
a) A NAT’s hairpinning behavior MUST be of type "External source IP address and port".

REQ-9: If a NAT translates TCP, it SHOULD translate ICMP Destination Unreachable (Type 3) messages.

REQ-10: Receipt of any sort of ICMP message MUST NOT terminate the NAT mapping or TCP connection for which the ICMP was generated.

9. Security Considerations

[BEHAVE-UDP] discusses security considerations for NATs that handle IP and unicast UDP traffic. Security concerns specific to handling TCP packets are discussed in this section.

Security considerations for REQ-1: This requirement does not introduce any TCP-specific security concerns.

Security considerations for REQ-2: This requirement does not introduce any TCP-specific security concerns. Simultaneous-open and other transitions in the TCP state machine are by-design and necessary for TCP to work correctly in all scenarios. Further, this requirement only affects connections already in progress as authorized by the NAT in accordance with its policy.

Security considerations for REQ-3: The security provided by the NAT is governed by its filtering behavior as addressed in [BEHAVE-UDP]. Connection-Dependent Filtering behavior is most secure from a firewall perspective, but severely restricts connection initiations through a NAT. Endpoint-Independent Filtering behavior, which is most transparent to applications, requires an attacker to guess the IP address and port of an active mapping in order to get his packet to an internal host. Address-Dependent Filtering, on the other hand, is less transparent than Endpoint-Independent Filtering but more transparent than Connection-Dependent Filtering; it is more secure than Endpoint-Independent Filtering as it requires an attacker to additionally guess the address of the external endpoint for a NAT session associated with the mapping and be able to receive packets addressed to the same. While this protects against most attackers on the Internet, it does not necessarily protect against attacks that originate from behind a remote NAT with a single IP address that is also translating a legitimate connection to the victim.

Security considerations for REQ-4: This document recommends that a NAT respond to unsolicited inbound SYN packets with an ICMP error delayed by a few seconds. Doing so may reveal the presence of a NAT to an external attacker. Silently dropping the SYN makes it harder to diagnose network problems and forces applications to wait for the TCP stack to finish several retransmissions before reporting an error. An implementer must therefore understand and carefully weigh the effects of not sending an ICMP error or rate-limiting such ICMP errors to a very small number.

Security considerations for REQ-5: This document recommends that a NAT that passively monitors TCP state keep idle sessions alive for at least 2 hours 4 minutes or 4 minutes depending on the state of the connection. If a NAT is under attack, it may attempt to actively determine the liveliness of a TCP connection or let the NAT administrator configure more conservative timeouts.

Security considerations for REQ-6: This requirement does not introduce any TCP-specific security concerns.

Security considerations for REQ-7: This requirement does not introduce any TCP-specific security concerns.

Security considerations for REQ-8: This requirement does not introduce any TCP-specific security concerns.

Security considerations for REQ-9: This requirement does not introduce any TCP-specific security concerns.

Security considerations for REQ-10: This requirement does not introduce any TCP-specific security concerns.

NAT implementations that modify TCP sequence numbers (e.g., for privacy reasons or for ALG support) must ensure that TCP packets with Selective Acknowledgement (SACK) notifications [RFC2018] are properly handled.

NAT implementations that modify local state based on TCP flags in packets must ensure that out-of-window TCP packets are properly handled. [RFC4953] summarizes and discusses a variety of solutions designed to prevent attackers from affecting TCP connections.

10. Acknowledgments

Joe Touch contributed the mechanism for handling unsolicited inbound SYNs. Thanks to Mark Allman, Francois Audet, Lars Eggert, Paul Francis, Fernando Gont, Sam Hartman, Paul Hoffman, Dave Hudson, Cullen Jennings, Philip Matthews, Tom Petch, Magnus Westerlund, and Dan Wing for their many contributions, comments, and suggestions.

11. References

11.1. Normative References

[BEHAVE-UDP] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

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

11.2. Informational References

[BEHAVE-ICMP] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP protocol", Work in Progress, June 2008.

[NAT-MIB] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.

[NATBLASTER] Biggadike, A., Ferullo, D., Wilson, G., and A. Perrig, "NATBLASTER: Establishing TCP connections between hosts behind NATs", Proceedings of the ACM SIGCOMM Asia Workshop (Beijing, China), April 2005.

[P2PNAT] Ford, B., Srisuresh, P., and D. Kegel, "Peer-to-peer communication across network address translators", Proceedings of the USENIX Annual Technical Conference (Anaheim, CA), April 2005.

[RFC1337] Braden, B., "TIME-WAIT Assassination Hazards in TCP", RFC 1337, May 1992.

[RFC1644] Braden, B., "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

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

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

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

[RFC4614] Duke, M., Braden, R., Eddy, W., and E. Blanton, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 4614, September 2006.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, July 2007.

[STUNT] Guha, S. and P. Francis, "NUTSS: A SIP based approach to UDP and TCP connectivity", Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture (Portland, OR), August 2004.

[TCPTRAV] Guha, S. and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls", Proceedings of the Internet Measurement Conference (Berkeley, CA), October 2005.

Authors' Addresses

Saikat Guha (editor)
Cornell University
331 Upson Hall
Ithaca, NY 14853
US
Phone: +1 607 255 1008
EMail: saikat@cs.cornell.edu

Kaushik Biswas
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
US
Phone: +1 408 525 5134
EMail: kbiswas@cisco.com

Bryan Ford
Max Planck Institute for Software Systems
Campus Building E1 4
D-66123 Saarbruecken
Germany
Phone: +49-681-9325657
EMail: baford@mpi-sws.org

Senthil Sivakumar
Cisco Systems, Inc.
7100-8 Kit Creek Road
PO Box 14987
Research Triangle Park, NC 27709-4987
US
Phone: +1 919 392 5158
EMail: ssenthil@cisco.com

Pyda Srisuresh
Kazeon Systems, Inc.
1161 San Antonio Rd.
Mountain View, CA 94043
US
Phone: +1 408 836 4773
EMail: srisuresh@yahoo.com

Full Copyright Statement

Intellectual Property