### Reducing the TIME-WAIT State Using TCP Timestamps

Abstract

   This document describes an algorithm for processing incoming SYN
   segments that allows higher connection-establishment rates between
   any two TCP endpoints when a TCP Timestamps option is present in the
   incoming SYN segment.  This document only modifies processing of SYN
   segments received for connections in the TIME-WAIT state; processing
   in all other states is unchanged.

Table of Contents

## [1](#).  Introduction

   The Timestamps option, specified in [RFC 1323](#) [[RFC1323](#)], allows a TCP
   to include a timestamp value in its segments that can be used to
   perform two functions: Round-Trip Time Measurement (RTTM) and
   Protection Against Wrapped Sequences (PAWS).

   For the purpose of PAWS, the timestamps sent on a connection are
   required to be monotonically increasing.  While there is no
   requirement that timestamps are monotonically increasing across TCP
   connections, the generation of timestamps such that they are
   monotonically increasing across connections between the same two
   endpoints allows the use of timestamps for improving the handling of
   SYN segments that are received while the corresponding four-tuple is
   in the TIME-WAIT state.  That is, the Timestamps option could be used
   to perform heuristics to determine whether to allow the creation of a
   new incarnation of a connection that is in the TIME-WAIT state.

This use of TCP timestamps is simply an extrapolation of the use of
Initial Sequence Numbers (ISNs) for the same purpose, as allowed by
RFC 1122 [RFC1122], and it has been incorporated in a number of TCP
implementations, such as that included in the Linux kernel [Linux].

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

## 2.  Improved Processing of Incoming Connection Requests

In a number of scenarios, a socket pair may need to be reused while
the corresponding four-tuple is still in the TIME-WAIT state in a
remote TCP peer.  For example, a client accessing some service on a
host may try to create a new incarnation of a previous connection,
while the corresponding four-tuple is still in the TIME-WAIT state at
the remote TCP peer (the server).  This may happen if the ephemeral
port numbers are being reused too quickly, either because of a bad
policy of selection of ephemeral ports, or simply because of a high
connection rate to the corresponding service.  In such scenarios, the
establishment of new connections that reuse a four-tuple that is in
the TIME-WAIT state would fail.  This problem is discussed in detail
in [INFOCOM-99].

In order to avoid this problem, Section 4.2.2.13 of RFC 1122
[RFC1122] states that when a connection request is received with a
four-tuple that is in the TIME-WAIT state, the connection request may
be accepted if the sequence number of the incoming SYN segment is
greater than the last sequence number seen on the previous
incarnation of the connection (for that direction of the data
transfer).  The goal of this requirement is to prevent the overlap of
the sequence number spaces of the old and new incarnations of the
connection so that segments from the old incarnation are not accepted
as valid by the new incarnation.

The same policy may be extrapolated to TCP timestamps.  That is, when
a connection request is received with a four-tuple that is in the
TIME-WAIT state, the connection request could be accepted if the
timestamp of the incoming SYN segment is greater than the last
timestamp seen on the previous incarnation of the connection (for
that direction of the data transfer).

The following paragraphs summarize the processing of SYN segments
received for connections in the TIME-WAIT state.  The processing of
SYN segments received for connections in all other states is
unchanged.  Both the ISN (Initial Sequence Number) and the Timestamps

option (if present) of the incoming SYN segment are included in the
heuristics performed for allowing a high connection-establishment
rate.

Processing of SYN segments received for connections in the TIME-WAIT
state SHOULD occur as follows:

o  If the previous incarnation of the connection used Timestamps,
   then:

   *  If TCP Timestamps would be enabled for the new incarnation of
      the connection, and the timestamp contained in the incoming SYN
      segment is greater than the last timestamp seen on the previous
      incarnation of the connection (for that direction of the data
      transfer), honor the connection request (creating a connection
      in the SYN-RECEIVED state).

   *  If TCP Timestamps would be enabled for the new incarnation of
      the connection, the timestamp contained in the incoming SYN
      segment is equal to the last timestamp seen on the previous
      incarnation of the connection (for that direction of the data
      transfer), and the Sequence Number of the incoming SYN segment
      is greater than the last sequence number seen on the previous
      incarnation of the connection (for that direction of the data
      transfer), honor the connection request (creating a connection
      in the SYN-RECEIVED state).

   *  If TCP Timestamps would not be enabled for the new incarnation
      of the connection, but the Sequence Number of the incoming SYN
      segment is greater than the last sequence number seen on the
      previous incarnation of the connection (for the same direction
      of the data transfer), honor the connection request (creating a
      connection in the SYN-RECEIVED state).

   *  Otherwise, silently drop the incoming SYN segment, thus leaving
      the previous incarnation of the connection in the TIME-WAIT
      state.

o  If the previous incarnation of the connection did not use
   Timestamps, then:

   *  If TCP Timestamps would be enabled for the new incarnation of
      the connection, honor the incoming connection request (creating
      a connection in the SYN-RECEIVED state).

     *  If TCP Timestamps would not be enabled for the new incarnation
        of the connection, but the Sequence Number of the incoming SYN
        segment is greater than the last sequence number seen on the
        previous incarnation of the connection (for the same direction
        of the data transfer), honor the incoming connection request
        (creating a connection in the SYN-RECEIVED state).

     *  Otherwise, silently drop the incoming SYN segment, thus leaving
        the previous incarnation of the connection in the TIME-WAIT
        state.

   Note:

      In the above explanation, the phrase "TCP Timestamps would be
      enabled for the new incarnation for the connection" means that the
      incoming SYN segment contains a TCP Timestamps option (i.e., the
      client has enabled TCP Timestamps), and that the SYN/ACK segment
      that would be sent in response to it would also contain a
      Timestamps option (i.e., the server has enabled TCP Timestamps).
      In such a scenario, TCP Timestamps would be enabled for the new
      incarnation of the connection.

      The "last sequence number seen on the previous incarnation of the
      connection (for the same direction of the data transfer)" refers
      to the last sequence number used by the previous incarnation of
      the connection (for the same direction of the data transfer), and
      not to the last value seen in the Sequence Number field of the
      corresponding segments.  That is, it refers to the sequence number
      corresponding to the FIN flag of the previous incarnation of the
      connection, for that direction of the data transfer.

   Many implementations do not include the TCP Timestamps option when
   performing the above heuristics, thus imposing stricter constraints
   on the generation of Initial Sequence Numbers, the average data
   transfer rate of the connections, and the amount of data transferred
   with them.  RFC 793 [RFC0793] states that the ISN generator should be
   incremented roughly once every four microseconds (i.e., roughly
   250,000 times per second).  As a result, any connection that
   transfers more than 250,000 bytes of data at more than 250 kilobytes/
   second could lead to scenarios in which the last sequence number seen
   on a connection that moves into the TIME-WAIT state is still greater
   than the sequence number of an incoming SYN segment that aims at
   creating a new incarnation of the same connection.  In those
   scenarios, the ISN heuristics would fail, and therefore the
   connection request would usually time out.  By including the TCP
   Timestamps option in the heuristics described above, all these
   constraints are greatly relaxed.

It is clear that the use of TCP timestamps for the heuristics
described above benefit from timestamps that are monotonically
increasing across connections between the same two TCP endpoints.

Note:
   The upcoming revision of RFC 1323, [1323bis], recommends the
   selection of timestamps such that they are monotonically
   increasing across connections.  An example of such a timestamp
   generation scheme can be found in [TS-Generation].

## 3.  Interaction with Various Timestamp Generation Algorithms

The algorithm proposed in Section 2 clearly benefits from timestamps
that are monotonically increasing across connections to the same
endpoint.  In particular, generation of timestamps such that they are
monotonically increasing is important for TCP instances that perform
the active open, as those are the timestamps that will be used for
the proposed algorithm.

While monotonically increasing timestamps ensure that the proposed
algorithm will be able to reduce the TIME-WAIT state of a previous
incarnation of a connection, implementation of the algorithm (by
itself) does not imply a requirement on the timestamp generation
algorithm of other TCP implementations.

In the worst-case scenario, an incoming SYN corresponding to a new
incarnation of a connection in the TIME-WAIT contains a timestamp
that is smaller than the last timestamp seen on the previous
incarnation of the connection, the heuristics fail, and the result is
no worse than the current state of affairs.  That is, the SYN segment
is ignored (as specified in [RFC1337]), and thus the connection
request times out, or is accepted after future retransmissions of the
SYN.

Some stacks may implement timestamp generation algorithms that do not
lead to monotonically increasing timestamps across connections with
the same remote endpoint.  An example of such algorithms is the one
described in [RFC4987] and [Opperman], which allows the
implementation of extended TCP SYN cookies.

Note:
   It should be noted that the "extended TCP SYN cookies" could
   coexist with an algorithm for generating timestamps such that they
   are monotonically increasing.  Monotonically increasing timestamps
   could be generated for TCP instances that perform the active open,
   while timestamps for TCP instances that perform the passive open
   could be generated according to [Opperman].

Some stacks (notably OpenBSD) implement timestamp randomization
algorithms which do not result in monotonically increasing ISNs
across connections.  As noted in [Silbersack], such randomization
schemes may prevent the mechanism proposed in this document from
recycling connections that are in the TIME-WAIT state.  However, as
noted earlier in this section in the worst-case scenario, the
heuristics fail, and the result is no worse than the current state of
affairs.

## 4.  Interaction with Various ISN Generation Algorithms

[RFC0793] suggests that the ISNs of TCP connections be generated from
a global timer, such that they are monotonically increasing across
connections.  However, this ISN-generation scheme leads to
predictable ISNs, which have well-known security implications
[CPNI-TCP].  [RFC1948] proposes an alternative ISN-generation scheme
that results in monotonically increasing ISNs across connections that
are not easily predictable by an off-path attacker.

Some stacks (notably OpenBSD) implement ISN randomization algorithms
which do not result in monotonically increasing ISNs across
connections.  As noted in [Silbersack], such ISN randomization
schemes break BSD's improved handling of SYN segments received for
connections that are in the TIME-WAIT state.

An implementation of the mechanism proposed in this document would
enable recycling of the TIME-WAIT state even in the presence of ISNs
that are not monotonically increasing across connections, except when
the timestamp contained in the incoming SYN is equal to the last
timestamp seen on the connection in the TIME-WAIT state (for that
direction of the data transfer).

## 5.  Security Considerations

[TCP-Security] contains a detailed discussion of the security
implications of TCP Timestamps and of different timestamp generation
algorithms.

## 6.  Acknowledgements

This document is based on part of the contents of the technical
report "Security Assessment of the Transmission Control Protocol
(TCP)" [CPNI-TCP] written by Fernando Gont on behalf of the United
Kingdom's Centre for the Protection of National Infrastructure (UK
CPNI).

## 7.  References

### 7.1.  Normative References

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

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

   [RFC1323]       Jacobson, V., Braden, B., and D. Borman, "TCP
                   Extensions for High Performance", RFC 1323,
                   May 1992.

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

### 7.2.  Informative References

   [1323bis]       Borman, D., Braden, R., and V. Jacobson, "TCP
                   Extensions for High Performance", Work in Progress,
                   March 2009.

   [CPNI-TCP]      CPNI, "Security Assessment of the Transmission
                   Control Protocol (TCP)", 2009,
                   <http://www.cpni.gov.uk/Docs/
                   tn-03-09-security-assessment-TCP.pdf>.

   [INFOCOM-99]    Faber, T., Touch, J., and W. Yue, "The TIME-WAIT
                   state in TCP and Its Effect on Busy Servers", Proc.
                   IEEE Infocom, 1999, pp. 1573-1583.

   [Linux]           Linux Kernel Organization, "The Linux Kernel
                     Archives", <http://www.kernel.org>.

   [Opperman]        Oppermann, A., "FYI: Extended TCP syncookies in
                     FreeBSD-current", post to the tcpm mailing list,
                     September 2006, <http://www.ietf.org/mail-archive/
                     web/tcpm/current/msg02251.html>.

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

   [RFC1948]         Bellovin, S., "Defending Against Sequence Number
                     Attacks", RFC 1948, May 1996.

   [RFC4987]         Eddy, W., "TCP SYN Flooding Attacks and Common
                     Mitigations", RFC 4987, August 2007.

   [Silbersack]      Silbersack, M., "Improving TCP/IP security through
                     randomization without sacrificing interoperability",
                     EuroBSDCon 2005.

   [TCP-Security]    Gont, F., "Security Assessment of the Transmission
                     Control Protocol (TCP)", Work in Progress,
                     January 2011.

   [TS-Generation]   Gont, F. and A. Oppermann, "On the generation of TCP
                     timestamps", Work in Progress, June 2010.

**Appendix A**.  **Behavior of the Proposed Mechanism in Specific Scenarios**

**A.1**.  **Connection Request after System Reboot**

   This section clarifies how this algorithm would operate in case a
   computer reboots, keeps the same IP address, loses memory of the
   previous timestamps, and then tries to reestablish a previous
   connection.

   Firstly, as specified in [RFC0793], hosts must not establish new
   connections for a period of 2*MSL (Maximum Segment Lifetime) after
   they boot (this is the "quiet time" concept).  As a result, in terms
   of specifications, this scenario should never occur.

   If a host does not comply with the "quiet time concept", a connection
   request might be sent to a remote host while there is a previous
   incarnation of the same connection in the TIME-WAIT state at the
   remote host.  In such a scenario, as a result of having lost memory
   of previous timestamps, the resulting timestamps might not be
   monotonically increasing, and hence the proposed algorithm might be
   unable to recycle the previous incarnation of the connection that is
   in the TIME-WAIT state.  This case corresponds to the current state
   of affairs without the algorithm proposed in this document.

Author's Address

   Fernando Gont
   UK Centre for the Protection of National Infrastructure

   EMail: fernando@gont.com.ar
   URI:   http://www.cpni.gov.uk