TCP Maintenance and Minor
Extensions (tcpm)
Internet-Draft
Intended status: BCP
Expires: December 12, 2010

F. Gont
UK CPNI

June 10, 2010

Reducing the TIME-WAIT state using TCP timestamps
draft-ietf-tcpm-tcp-timestamps-00.txt

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.

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 December 12, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

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 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 timestamp 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.

In order to avoid this problem, RFC 1122 [RFC1122] (in Section 4.2.2.13) states that 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 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). This requirement aims at avoiding the sequence number space of the new and old incarnations of the connection to overlap, thus avoiding old segments from the previous incarnation of the connection to be accepted as valid by the new connection.

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. Both the ISN (Initial Sequence Number) and the timestamp 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 synchronized states should occur as follows:

- If a SYN segment is received for a connection in any synchronized state other than TIME-WAIT, respond with an ACK, applying rate-throttling.

- If the corresponding connection is in the TIME-WAIT state, then,

  * 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), honour 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 larger than the last sequence number seen on the previous incarnation of the connection (for that direction of the data transfer), then honour 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 larger than the last sequence number seen on the previous incarnation of the connection (for the same direction of the data transfer), honour 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.

* 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, honour the incoming connection request.

+ If TCP timestamps would not be enabled for the new incarnation of the connection, but the Sequence Number of the incoming SYN segment is larger than the last sequence number seen on the previous incarnation of the connection

(for the same direction of the data transfer), then honour the incoming connection request (even if the sequence number of the incoming SYN segment falls within the receive window of the previous incarnation of the connection).

+ 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 timestamp 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 250000 times per second). As a result, any connection that transfers more than 250000 bytes of data at more than 250 KB/s 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 4.4BSD

heuristics would fail, and therefore the connection request would usually time out. By including the TCP timestamp 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.

3. Interaction with various timestamps generation algorithms

The algorithm proposed in Section 2 clearly benefits of timestamps that are monotonically-increasing across connections to the same end-point. In particular, generation of timestamps such that they are monotonically-increasing timestamps are important for TCPs 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 does not imply by itself a requirement on the timestamps generation algorithm of other TCPs.

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 TIME_WAIT state is assassinated, with the connection request being rejected (as specified in [RFC0793]), or,
- 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 timestamps 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], that allows the implementation of extended TCP SYN cookies.

Note:
It should be noted that this algorithm could co-exist with an algorithm for generating timestamps such that they are monotonically-increasing. Monotonically increasing timestamps could be generated for TCPs that perform the active open, while timestamps for TCPs that perform the passive open could be generated according to [Opperman].

4. Corner-cases

4.1. Connection request after system reboot

The question was raised on the tcpm mailing-list as to how this algorithm would operate in case a computer reboots, keeps the same IP address, looses memory of the previous time stamps, 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 after they boot (this is the "quiet time" concept). As a result, specs-wise, this scenario should never occur.

If a host does not comply with the "quiet time concept", then the possible scenarios are:

- If the selected timestamp for the new connection is monotonically-increasing with respect to the last timestamp seen on the previous incarnation of the connection, the TIME-WAIT state is tossed, and the new connection request succeeds.

- Otherwise, the connection request may time out or be rejected (depending on whether the workaround described in [RFC1337] is implemented or not). This case corresponds to the current state-of-affairs without the algorithm proposed in this document.

5. Security Considerations

While the algorithm described in this document for processing incoming SYN segments would benefit from TCP timestamps that are monotonically-increasing across connections, this document does not propose any specific algorithm for generating timestamps, nor does it require monotonically-increasing timestamps across conenctions.

[CPNI-TCP] contains a detailed discussion of the security implications of TCP timestamps.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The author of this document would like to thank (in alphabetical order) Mark Allman, Christian huitema, Alfred Hoenes, Eric Rescorla, Joe Touch, and Alexander Zimmermann for providing valuable feedback on an earlier version of this document.

Additionally, the author would like to thank David Borman for a fruitful discussion on TCP timestamps at IETF 73.

Finally, the author would like to thank the United Kingdom's Centre for the Protection of National Infrastructure (UK CPNI) for their continued support.

8. References

8.1. 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.

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

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

8.2. Informative References

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

[Linux] The Linux Project, "http://www.kernel.org".

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

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

Appendix A. Changes from previous versions of the draft (to be removed by the RFC Editor before publishing this document as an RFC)

A.1. Changes from draft-gont-tcpm-tcp-timestamps-04

- Draft resubmitted as draft-ietf.

A.2. Changes from draft-gont-tcpm-tcp-timestamps-03

- Changed the document title
- Removed all the text related to the algorithm earlier proposed for timestamps generation.
- Addresses comments received from Alexander Zimmermann, Christian Huitema, Joe Touch, and others.

A.3. Changes from draft-gont-tcpm-tcp-timestamps-02

- Minor edits (the I-D was just about to expire, so it was resubmitted with almost no changes).

A.4. Changes from draft-gont-tcpm-tcp-timestamps-01

- Version -01 of the draft had expired, and hence the I-D is resubmitted to make it available again (no changes).

A.5. Changes from draft-gont-tcpm-tcp-timestamps-00

- Fixed author's affiliation.

- Addressed feedback submitted by Alfred Hoenes (see: http://www.ietf.org/mail-archive/web/tcpm/current/msg04281.html), plus nits sent by Alfred off-list.

Author's Address

Fernando Gont
UK Centre for the Protection of National Infrastructure

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