TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Intended status: BCP
Expires: August 6, 2009

F. Gont
UK CPNI

February 2, 2009

On the generation of TCP timestamps
draft-gont-tcpm-tcp-timestamps-01.txt

Status of this Memo

This Internet-Draft will expire on August 6, 2009.

Copyright Notice

Abstract

This document describes an algorithm for selecting the timestamps (TS value) used for TCP connections that use the TCP timestamp option, such that the resulting timestamps are monotonically-increasing across connections that involve the same four-tuple {local IP address, local TCP port, remote IP address, remote TCP port}. The properties of the algorithm are such that it reduces the possibility of an attacker of guessing the exact value. Additionally, it 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.

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].

In order to avoid the security implications of predictable timestamps, the proposed algorithm generates timestamps such that the possibility of an attacker guessing the exact value is reduced.

Section 2 proposes the aforementioned algorithm for generating TCP timestamps. Section 3 describes an improved processing of incoming connection requests, that may allow higher connection-establishment rates to any TCP end-point.

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. Proposed algorithm

It is RECOMMENDED that timestamps are generated with a similar algorithm to that introduced by RFC 1948 [RFC1948] for the generation of Initial Sequence Numbers (ISNs). That is,

timestamp = T() + F(localhost, localport, remotehost, remoteport, secret_key)

where the result of T() is a global system clock that complies with the requirements of Section 4.2.2 of RFC 1323 [RFC1323], and F() is a

function that should not be computable from the outside without knowledge of the secret key (secret_key). Therefore, we suggest F() to be a cryptographic hash function of the connection-id and some secret data (which could be chosen randomly).

F() provides an offset that will be the same for all incarnations of a connection between the same two endpoints, while T() provides the monotonically increasing values that are needed for PAWS.

3. 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 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 depends on the timestamps to be monotonically increasing across connections between the same two TCP endpoints. Therefore, we strongly advice to generate timestamps as described in Section 2.

4. Security Considerations

This document describes an algorithm that can be used to obfuscate the timestamp value used for new connections, such that the possibility of an attacker guessing the exact value is reduced.

Some implementations are known to maintain a global timestamp clock, which is used for all connections. This is undesirable, as an attacker that can establish a connection with a host would learn the timestamp used for all the other connections maintained by that host, which could be useful for performing any attacks that require the attacker to forge TCP segments. Some implementations are known to initialize their global timestamp clock to zero when the system is bootstrapped. This is undesirable, as the timestamp clock would disclose the system uptime.

The algorithm discussed in this document for generating the TCP timestamps avoids these problems by generating timestamps as monotonically-increasing function with a per-connection-id random offset. [CPNI-TCP]

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgements

The author of this document would like to thank (in alphabetical order) Alfred Hoenes and Eric Rescorla 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.

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

[CPNI-TCP] CPNI, "Security Assessment of the Transmission Control Protocol (TCP)", (to be published) .

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

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

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-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