TCP Maintenance and Minor
Extensions Working Group
Internet-Draft
Intended status: Informational
Expires: August 18, 2011

M. Bashyam
Ocarina Networks, Inc
M. Jethanandani
A. Ramaiah
Cisco

February 14, 2011

Clarification of sender behavior in persist condition. draft-ietf-tcpm-persist-02.txt

Abstract

This document clarifies the Zero Window Probes (ZWP) described in [RFC1122]. In particular, it clarifies the actions that can be taken on connections which are experiencing the ZWP condition.

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 August 18, 2011.

Copyright Notice

Table of Contents

1. Introduction

Section 4.2.2.17 of [RFC1122] says:

> "A TCP MAY keep its offered receive window closed indefinitely. As long as the receiving TCP continues to send acknowledgments in response to the probe segments, the sending TCP MUST allow the connection to stay open."
>
> DISCUSSION:
>
> It is extremely important to remember that ACK (acknowledgment) segments that contain no data are not reliably transmitted by TCP.

Therefore zero window probing SHOULD be supported to prevent a connection from hanging forever if ACK segments that re-opens the window is lost. The condition where the sender goes into the Zero-Window Probe (ZWP) mode is typically known as the 'persist condition'.

This guidance is not intended to preclude resource management by the operating system or application, which may request connections to be aborted regardless of them being in the persist condition, and the TCP implementation should, of course, comply by aborting such connections. TCP implementations strictly adhering to Section 4.2.2.17 of [RFC1122] have the potential to make systems vulnerable to Denial of Service (DoS) scenarios where attackers tie up resources by keeping connections in the persist condition, if such resource management is not performed external to the protocol implementation.

Section 2 of this document describes why implementations must not close connections merely because they are in the persist condition, yet must still allow such connections to be closed on command. Section 3 outlines a simple attack on systems that do not sufficiently manage connections in this state. Section 4 concludes

with a requirements-language clarification to the RFC 1122 requirement.

2. Discussion on RFC 1122 Requirement

Per [RFC1122] as long as the ACK's are being received for window probes, a connection can continue to stay in the persist condition. This is an important feature because typically applications would want the TCP connection to stay open unless an application explicitly closes the connection.

For example take the case of user running a network print job during which the printer runs out of paper and is waiting for the user intervention to reload the paper tray. The printer may not be reading data from the printing application during this time. Although this may result in a prolonged ZWP state, it would be premature for TCP to take action on its own and close the printer connecting merely due to its lack of progress. Once the printer's paper tray is reloaded (which may be minutes, hours, or days later), the print job should be able to continue uninterrupted over the same TCP connection.

Systems that adhere too strictly to the above verbiage of [RFC1122] may fall victim to DoS attacks, by not supporting sufficient mechanisms to allow release of system resources tied up by connections in the persist condition during times of resource exhaustion. For example, if we take the case of a busy server where multiple (attacker) clients can advertise a zero window forever (by reliably acknowledging the ZWPs). This could eventually lead to the resource exhaustion in the server system. In such cases the application or operating system would need to take appropriate action on the TCP connection to reclaim their resources and continue to

persist legitimate connections.

The problem is applicable to TCP and TCP derived flow-controlled transport protocols like SCTP.

Clearly, a system should be robust to such attacks and allow connections in the persist condition to be aborted in the same way as any other connection. Section 4 of this document provides the requisite clarification, in standards language, to permit such resource management

3. Description of one Simple Attack

To illustrate a potential DoS scenario, consider the case where many client applications open TCP connection with a HTTP [RFC2616] server, and each sends a GET request for a large page and stops reading the response partway through. This causes the client's TCP implementation to advertise a zero window to the server. For every large HTTP response, the server is left holding on to the response data in its sending queue. The amount of response data held will depend on the size of the send buffer and the advertised window. If the clients never read the data in their receive queues in order to clear the persist condition, the server will continue to hold that data indefinitely. Since there may be a limit to the operating system kernel memory available for TCP buffers, this may result in DoS to legitimate connections by locking up the necessary resources. If the above scenario persists for an extended period of time, it will lead to TCP buffers and connection blocks starvation causing legitimate existing connections and new connection attempts to fail.

A clever application might detect such attacks with connections that are not making progress, and could close these connections. However, some applications might have transferred all the data to the TCP

socket and subsequently closed the socket leaving the connection with no controlling process, hereby referred to as orphaned connections. Such orphaned connections might be left holding the data indefinitely in their sending queue.

CERT has released an advisory in this regard[VU723308] and is making vendors aware of this DoS scenario.

Appendix A of this document provides a simple mitigation to such attacks. More sophisticated attacks are possible which can build on this vulnerability and may remain effective even when mitigated with the mechanism prescribed in Appendix A of this document.

4. Clarification Regarding RFC 1122 Requirements

As stated in [RFC1122], a TCP implementation MUST NOT close a connection merely because it seems to be stuck in the ZWP or persist condition. Unstated in RFC 1122, but implicit for system robustness, a TCP implementation MUST allow connections in the ZWP or persist condition to be closed or aborted by their applications or other resource management routines in the operating system.

In order to provide some level of robustness to DoS attacks, a TCP implementation MAY provide a feedback regarding the persist condition to the application if requested to do so or an application or other resource manager can query the health of the TCP connection allowing it to take the desired action. All such techniques are in complete compliance of [RFC0793] and [RFC1122].

5. Acknowledgments

This document was inspired by the recent discussions that took place regarding the TCP persist condition issue in the TCPM WG mailing list [TCPM]. The outcome of those discussions was to come up with a draft that would clarify the intentions of the ZWP referred by RFC 1122. We would like to thank Mark Allman, Ted Faber and David Borman for clarifying the objective behind this draft. To Wesley Eddy for his

extensive editorial comments and to Dan Wing, Mark Allman and Fernando Gont on providing feedback on the document.

6. Appendix A, Programming Considerations

As a potential implementation guideline, the authors are documenting some of the programming considerations. This should not be in any way construed as the only way that the mitigation against the DoS condition can be achieved. Applications can choose their own implementations on how to deal with this DoS scenario, and should be aware that this mitigation is only effective at combating the simple attack scenario described in this document, and does not handle even slightly more sophisticated attacks based on the same or similar concepts.

Note, this persist condition is mutually exclusive from a persist condition where we are not getting zero windows acknowledgement for the probes.

The technique described here allows an application to specify to the operating system that it consents to aborting such connections. Implementers can choose to in addition provide an asynchronous notification interface to inform the application of the connection in the persist condition, if they want the application to abort the connection. In the case where the application has terminated or orphaned the connection, the TCP or kernel code will go ahead and clear the connection and reclaim its resources.

The key consideration in putting a solution together is to be able to detect a connection that is in persist condition. The application through the socket interface will be able to inform TCP implementation or kernel of how long they are willing to have connections wait in the persist condition.

PERSIST_TIMEOUT

Format:

int setsockopt (sockfd, SOL_TCP, SO_PERSISTTIMEO,
persist_timeout_value, length)

int getsockopt (sockfd, SOL_TCP, SO_PERSISTTIMEO,
persist_timeout_value, length)

where persist_timeout_value recorded in seconds is of type int, the length is set to four.

The above interface allows applications to inform TCP what to do when the local connection stays in the persist condition. Note that the default value of persist_timeout_value is -1 which implies it is infinite.

TCP sender will save the current time in the connection block when it receives a zero window ACK. This time is referred to as the persist entry time. Thereafter every time the probe timer expires and before it sends another probe or an ACK carrying zero window is received a check will be done to see how long the connection has been in persist condition by comparing the current time to the persist entry time. If the timeout has been exceeded, the connection will be aborted.

Any time a ACK is received that advertises a non-zero window, the persist entry time is cleared to take the connection out of the persist condition.

7. Informative 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.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[TCPM] TCPM, "IETF TCPM Working Group and mailing list http://www.ietf.org/html.charters/tcpm.charter.html".

[VU723308] Manion, "Vulnerability in Web Servers http://www.kb.cert.org/vuls/id/723308", July 2009.

Authors' Addresses

Murali Bashyam
Ocarina Networks, Inc
42 Airport Parkway
San Jose, CA 95110
USA

Phone: +1 (408) 512-2966
Email: mbashyam@ocarinanetworks.com

Mahesh Jethanandani
Cisco
170 Tasman Drive
San Jose, CA 95134
USA

Phone: +1 (408) 527-8230
Email: mahesh@cisco.com

Anantha Ramaiah
Cisco
170 Tasman Drive
San Jose, CA 95134
USA

Phone: +1 (408) 525-6486
Email: ananth@cisco.com