TCP Maintenance and Minor Extensions Working Group
Internet-Draft
Intended status: Informational
Expires: May 14, 2011

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

November 10, 2010

Clarification of sender behaviour in persist condition.
draft-ietf-tcpm-persist-01.txt

Abstract

This document attempts to clarify the notion of the Zero Window Probes (ZWP) described in RFC 1122 [RFC1122]. In particular, it clarifies the actions that can be taken on connections which are experiencing the ZWP condition. The motivation for this document stems from the belief that TCP implementations strictly adhering to the current RFC language have the potential to become vulnerable to Denial of Service (DoS) scenarios.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Discussion on RFC 1122 Requirement 4
3. Description of Attack 5
4. Clarification Regarding RFC 1122 Requirements 6
5. Conclusion 7
6. Acknowledgments 8
7. Programming Considerations 9
8. Informative References 10
Authors' Addresses 11

1. Introduction

TCP implementations strictly adhering to Section 4.2.2.17 of [RFC1122] have the potential to become vulnerable to Denial of Service (DoS) scenarios. That section 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'. It is under this condition that the sending TCP can become vulnerable to DoS.

2. Discussion on RFC 1122 Requirement

It needs to be emphasised that TCP MUST NOT take any action of its own when a particular connection is in persist condition for a long time. As per RFC 1122 as long as the ACK's are being received for window probes, it can continue to stay in persist condition. This is important because typically applications would want the TCP connection to stay open unless it explicitly closes the connection. For example take the case of user running a print job and the printer ran out of paper waiting for the user intervention. It would be premature for TCP to take action on its own. Hence TCP cannot act as a resource manager and it is the system or application's responsibility to take appropriate action.

At the same time, many existing TCP implementations that adhere strictly to the above verbiage of RFC 1122 may fall victim to DOS attacks, if appropriate measures are not followed. For example, if we take the case of a busy server where multiple clients can advertise a zero forever (by reliably acknowledging the ZWP's), it could eventually lead to the resource exhaustion in the system. In such cases the system would need to take appropriate action on the TCP connection to reclaim the resources.

This document is not intended to provide any advice on any particular resource management scheme that can be implemented to circumvent DOS issues arising due to the connections stuck in the persist state.

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

In summary, TCP MUST NOT take any action on its own to abort a connection in persist condition. Applications however can request that a connection in persist condition be aborted. The resource manager in the operating system when faced with depleted resources can also ask TCP to abort a connection.

3. Description of Attack

If TCP implementations strictly follow RFC 1122 and there is no instruction on what to do in persist condition, connections will encounter an indefinite wait. To illustrate this, consider the case where the client application opens a TCP connection with a HTTP [RFC2616] server, sends a GET request for a large page and stops reading the response. This would cause the client TCP 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 send queue. The amount of response data held will depend on the size of the send buffer and the advertised window. If the client never reads the data in its receive queue or clears the persist condition, the server will continue to hold that data indefinitely. Multiple such TCP connections stuck in the same scenario on the server would cause resource depletion resulting in a DoS situation on the server.

Applications on the sender can transfer all the data to the TCP socket and subsequently close the socket leaving the connection with no controlling process, hereby referred to as orphaned connection. If the application on the receiver refuses to read the data, the orphaned connection will be left holding the data indefinitely in its send queue.

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.

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

4. Clarification Regarding RFC 1122 Requirements

A consequence of adhering to the above requirement mandated by RFC 1122 is that multiple TCP receivers advertising a zero window to a server could exhaust the connection and buffer resources of the sender. In such cases, and specially when the receiver is reliably acknowledging zero window probe, to achieve robustness, the system should be able to take appropriate action on those TCP connections and reclaim resources. A possible action could be to terminate the connection and such an action is in the spirit of RFC 1122.

In order to accomplish this action, TCP MAY provide a feedback regarding the persist condition to the application if requested to do so or the application or the resource manager can query the health of the TCP connection which would allow it to take the desired action. All such actions are in complete compliance of RFC 793 and RFC 1122.

5. Conclusion

The document addresses the fact that terminating TCP connections stuck in the persist condition does not violate RFC 1122 or RFC 793. It also suggests that TCP must not abort any connection until explicitly requested by the application or the operating system to do so. The potential implementation guidelines of the request and the action are documented in Section 7, and the details of mitigating the DoS attack are left to the implementer.

6. 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 and David Borman for clarifying the objective behind this draft. To Dan Wing, Mark Allman and Fernando Gont on providing feedback on the document.

7. 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 sceanrio.

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 can inform TCP or kernel of how long they are willing to wait in persist condition. When the connection reaches that particular timeout value a EPERSISTTIMEOUT notification will be sent to the application. The application on receiving the notification can turn around and issue a close. In the case, the application has terminated, TCP or kernel will go ahead and clear the connection and reclaim the resoruces. Note, this persist condition is mutually exclusive from a persist condition where we are not getting zero windows acknowledgement for the probes.

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 and the length is four.

The above interface allows applications to inform TCP that when the local connection stays in persist condition it can be aborted after a set time. Note that the default value of this option 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 persist condition.

8. 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 Systems
170 Tasman Drive
San Jose, CA 95134
USA

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

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

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