Network Working Group
Internet-Draft
Expires: October 18, 2004

R. Stewart
Editor

April 19, 2004

Transmission Control Protocol security considerations
draft-ietf-tcpm-tcpsecure-00.txt

Status of this Memo

Copyright Notice

Abstract

TCP (RFC793 [1]) is widely deployed and one of the most often used reliable end to end protocols for data communication. Yet when it was defined over 20 years ago the internet, as we know it, was a

different place lacking many of the threats that are now common. Recently several rather serious threats have been detailed that can pose new methods for both denial of service and possibly data injection by blind attackers. This document details those threats and also proposes some small changes to the way TCP handles inbound segments that either eliminate the threats or at least minimize them to a more acceptable level.

Table of Contents

1. Introduction

TCP (RFC793 [1]) is widely deployed and one of the most often used reliable end to end protocols for data communication. Yet when it was defined over 20 years ago the internet, as we know it, was a different place lacking many of the threats that are now common. Recently several rather serious threats have been detailed that can pose new methods for both denial of service and possibly data injection by blind attackers. This document details those threats and also proposes some small changes to the way TCP handles inbound segments that either eliminate the threats or at least minimize them to a more acceptable level.

Most of these changes violate some of the handling procedures for DATA, RST and SYN's as defined in RFC793 [1] but do not cause interoperability issues. The authors feel that many of the changes proposed in this document would, if TCP were being standardized today, be required to be in the base TCP document and the lack of these procedures is more an artifact of the time when TCP was developed than any strict requirement of the protocol.

2. Blind reset attack using the RST bit

2.1 Description of the attack

It has been traditionally thought that for a blind attacker to reset a TCP connection the attacker would have to guess a single sequence number in the TCP sequence space. This would in effect require an attacker to generate (2^^32/2) segments in order to reset a connection. Recent papers have shown this to not necessarily be the case. An attacker need only guess a number that lies between the last sequence number acknowledged and the last sequence number acknowledged added to the receiver window (RCV.WND). Modern operating systems normally default the RCV.WND to about 32,768 bytes. This means that a blind attacker need only guess 65,535 RST segments (2^^32/(RCV.WND*2)) in order to reset a connection. At DSL speeds this means that most connections (assuming the attacker can accurately guess both ports) can be reset in under 200 seconds (usually far less). With the rise of broadband availability and increasing available bandwidth, many Operating Systems have raised their default RCV.WND to as much as 64k, thus making these attacks even easier.

2.2 Solution

RFC793 [1] currently requires handling of a segment with the RST bit when in a synchronized state to be processed as follows:

1) If the RST bit is set and the sequence number is outside the expected window, silently drop the segment.
2) If the RST bit is set and the sequence number is acceptable i.e.: (RCV.NXT <= SEG.SEQ <= RCV.NXT+RCV.WND) then reset the connection.

Instead, the following changes should be made to provide some protection against such an attack.

A) If the RST bit is set and the sequence number is outside the expected window, silently drop the segment.
B) If the RST bit is exactly the next expected sequence number, reset the connection.
C) If the RST bit is set and the sequence number does not exactly match the next expected sequence value, yet is within the acceptable window (RCV.NXT < SEG.SEQ <= RCV.NXT+RCV.WND) send an acknowledgment.

This solution forms a challenge/response with any RST where the value does not exactly match the expected value and yet the RST is within the window. In cases of a legitimate reset without the exact sequence number, the consequences of this new challenge/response will be that the peer requires an extra round trip time before the connection can be reset.

3. Blind reset attack using the SYN bit

3.1 Description of the attack

The reset attack which uses the RST bit highlights another possible avenue for a blind attacker. Instead of using the RST bit an attacker can use the SYN bit as well to tear down a connection. Using the same guessing technique, repeated SYN's can be generated with sequence numbers incrementing by an amount not larger than the window size apart and thus eventually cause the connection to be terminated.

3.2 Solution

RFC793 [1] currently requires handling of a segment with the SYN bit set in the synchronized state to be as follows:

1) If the SYN bit is set and the sequence number is outside the expected window, send an ACK back to the sender.
2) If the SYN bit is set and the sequence number is acceptable i.e.: (RCV.NXT <= SEG.SEQ <= RCV.NXT+RCV.WND) then send a RST segment to the peer.

Instead, changing the handling of the SYN to the following will provide complete protection from this attack:

A) If the SYN bit is set and the sequence number is outside the expected window, send an ACK back to the peer.
B) If the SYN bit is set and the sequence number is an exact match to the next expected sequence (RCV.NXT == SEG.SEQ) then send an ACK segment to the peer but before sending the ACK segment subtract one from value being acknowledged.
C) If the SYN bit is set and the sequence number is acceptable i.e.: (RCV.NXT < SEG.SEQ <= RCV.NXT+RCV.WND) then send an ACK segment to the peer.

By always sending an ACK to the sender, a challenge/response is setup with the peer. A legitimate peer, after restart, would not have a TCB in the synchronized state. Thus when the ACK arrives the peer should send a RST segment. Note that for the case of an attacker sending a SYN that exactly matches the RCV.NXT value, by sending a ACK that is less than the RCV.NXT value the true peer will drop the ACK as an old duplicate. In cases where a valid restarting peer picks the ISS number to match the RCV.NXT, sending an ACK value one less than RCV.NXT will cause the restarted peer to see the ACK value as invalid

and thus send a RST.

4. Blind data injection attack

4.1 Description of the attack

A third type of attack is also highlighted by both reset attacks. It is quite possible to inject data into a TCP connection by simply guessing a sequence value that is within the window. The ACK value of any data segment is considered valid as long as it does not acknowledge data ahead of the next segment to send. In other words an ACK value is acceptable if it is (SND.UNA-(2^^31-1)) <= SEG.ACK < SND.NXT). This means that an attacker simply need guess two ACK values with every guessed sequence number so that the chances of successfully injecting data into a connection are 1 in ((2^^32/(RCV.WND * 2)) * (2^^32/(SND.WND * 2))).

4.2 Solution

An additional input check should be added to any incoming segment. The ACK value should be acceptable only if it is in the range of ((SND.UNA - MAX.SND.WND) <= SEG.ACK < SND.NXT). MAX.SND.WND is defined as the largest window that the local receiver has ever advertised to its peer. This window is the scaled value i.e. the value may be larger than 65,535 bytes. This small check will greatly reduce the vulnerability of an attacker guessing a valid sequence number since not only must he/she guess the sequence number in window, but must also guess a proper ack value within a scoped range. This solution reduces but does not eliminate the ability to generate false segments. It does however reduce the probability that invalid data will be injected to a more acceptable level when the maximum send and receive windows do not grow beyond 65,535 bytes. For those applications that wish to close this attack completely RFC2385 [2] should be deployed between the two endpoints.

5. Contributors

The following people worked under extreme pressure and short notice through the 2003 holiday's to come up with a set of solutions for these attacks. Their contributions and ideas on how to "fix" these TCP weaknesses are inter-mixed with each other to arrive at the set of solutions presented in this document. Shrirang Bage of Cisco Systems, Mark Baushke of Juniper Networks, Mitesh Dalal of Cisco Systems, Frank Kastenholz of Juniper Networks, Amol Khare of Cisco Systems, Qing Li of Wind River Systems Inc., Peter Lei of Cisco Systems, Paul Goyette of Juniper Networks, Patrick Mahan of Cisco Systems, Preety Puri of Wind River Systems Inc., Anantha Ramaiah of Cisco Systems, Art Stine of Juniper Networks, Xiaodan Tang of QNX, and David Wang of Juniper Networks.

6. Acknowledgments

Special thanks to Damir Rajnovic for suggestions and comments.

7. References

7.1 Normative References

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

7.2 Informative References

[2] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

Author's Address

Randall R. Stewart
Editor
8725 West Higgins Road
Suite 300
Chicago, IL 60631
USA

Phone: +1-815-477-2127
EMail: rrs@cisco.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.