TCPM
Internet-Draft
Intended status: Standards TrackCISPA Helmholtz Center for Information Security
Expires: 7 October 2026

Y. Pan
C. Rossow

5 April 2026

Improve TCP Handling of Out-of-Window Packets to Mitigate Ghost ACKs
draft-ietf-tcpm-tcp-ghost-acks-04

Abstract

Historically, TCP as specified in RFC 793 was threatened by the blind data injection attack because of the loose SEG.ACK value validation, where the SEG.ACK value of a TCP segment is considered valid as long as it does not acknowledge data ahead of what has been sent. RFC 5961 improved the input validation by shrinking the range of acceptable SEG.ACK values in a TCP segment. Later, RFC 9293 incorporated the updates proposed by RFC 5961 as a TCP stack implementation option.
However, an endpoint that follows the RFC 9293 specifications can still accept a TCP segment containing an SEG.ACK value acknowledging data that the endpoint has never sent. This document specifies small modifications to the way TCP verifies incoming TCP segments' SEG.ACK value to prevent TCP from accepting such invalid SEG.ACK values.

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 https://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 7 October 2026.

Copyright Notice

Table of Contents

1. Introduction

TCP as specified in [RFC0793] is widely deployed in today's Internet. Against the threat of the blind data injection attack, [RFC5961] section 5 proposed to improve the validation of the SEG.ACK field of incoming TCP segments. Currently, [RFC9293] is the latest main document for TCP, which obsoletes [RFC0793] and incorporates the SEG.ACK validation proposed by [RFC5961] as an optional implementation choice. The SEG.ACK validation introduced in [RFC9293] (with or without the [RFC5961] implementation choice) accepts a certain range of SEG.ACK values before SND.UNA as duplicate/old ACK values. This also applies to connections without data (or with little data) transferred previously. Consequently, current SEG.ACK validation accepts segments with invalid SEG.ACK values that acknowledge data that an endpoint has never sent as "duplicate/old" SEG.ACK values (ghost ACKs).
This document aims to improve the SEG.ACK value validation in [RFC9293], such that TCP would only accept duplicate/old SEG.ACK values acknowledging data already sent by the endpoint, eliminating the security risks imposed by ghost ACKs.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

TCP terminology should be interpreted as described in [RFC9293].

2. Ghost ACKs

As described in [RFC9293], when receiving a segment, the endpoint performs checks on the SEG.ACK field of the incoming segment. Suppose the TCP stack has implemented the mitigation for blind data injection attack proposed by Section 5 of [RFC5961], an incoming segment whose SEG.ACK value satisfies the condition SND.UNA - MAX.SND.WND =< SEG.ACK =< SND.NXT is considered acceptable, and the segment is further processed. When the [RFC5961] mitigation is not implemented, an incoming segment with SEG.ACK =< SND.NXT is accepted and further processed.

However, there are cases where the number of bytes sent by the endpoint is less than MAX.SND.WND or 2^31 - 1, and this can result in accepting a segment with an SEG.ACK value acknowledging bytes the endpoint has never sent.
As a concrete example, consider a newly established TCP connection without data transferred during the handshake. There is SND.UNA == SND.NXT == ISS + 1. In this case, any segments with SEG.ACK < SND.UNA acknowledges bytes that the endpoint has never sent, but they are still considered acceptable since they satisfy the above SEG.ACK validation condition.

3. Security Implications of Ghost ACKs

Ghost ACKs allow an attacker to inject payloads into a newly established connection. This extends the threat model as described in [RFC5961], where an off-path attacker can perform injection attacks against an existing foreign connection. Ghost ACKs further allow attackers that spoof the TCP handshake to use the spoofed TCP connection and transmit payloads [SP2024Spoof].

4. Mitigation

TCP stacks MAY implement one of the following two mitigations. Both mitigation options assume [RFC5961] is already supported by the TCP stack.

4.1. Mitigation Option 1 (generic)

TCP stacks that implement this mitigation SHOULD add the additional boolean state variable NO_ISS_CHECK for each established connection. This variable SHOULD be initialized to false. At the beginning of the SEG.ACK validation, it SHOULD be checked if the ISS is still needed:

```
if (!NO_ISS_CHECK && SND.UNA >= ISS + (65535 << Snd.Wind.Shift)) {
    /* Checking SEG.ACK against ISS is definitely redundant. */
    NO_ISS_CHECK = true;
}
```

Snd.Wind.Shift is defined in [RFC7323]. Then a local variable ACK.MIN SHOULD be computed, which is later used to validate the SEG.ACK. It is used to perform the validation, which is stricter.

```
if (NO_ISS_CHECK) {
    /* Check for too old ACKs (RFC 5961, Section 5.2). */
    ACK.MIN = SND.UNA - MAX.SND.WND;
} else {
    if (ISS + 1 > SND.UNA - MAX.SND.WND) {
        /* Checking for ghost ACKs is stricter. */
        ACK.MIN = ISS + 1;
    } else {
        /* Checking for too old ACKs (RFC 5961, Section 5.2) is stricter. */
        ACK.MIN = SND.UNA - MAX.SND.WND;
    }
}
```

Finally the validation of SEG.ACK SHOULD be performed:

```
if (SEG.ACK < ACK.MIN) {
      send_challenge_ack;
      return;
}
```

4.2. Mitigation Option 2 (assumes [RFC4898] support)

TCP stacks that implemented the [RFC4898] tcpEStatsAppHCThruOctetsAcked statistics, which tracks the number of bytes that are already acknowledged by the peer, can adopt this option.

To implement this mitigation, TCP stacks MUST add the SND.UNA - min(MAX.SND.WND, tcpEStatsAppHCThruOctetsAcked) =< SEG.ACK input checks for SEG.ACK values of any incoming segments. Segments with ACK values satisfying the above condition are further validated and

processed as specified in [RFC9293]. Otherwise, the segment MUST be discarded and a challenge ACK sent back. Though unlikely to happen, the 64-bit tcpEStatsAppHCThruOctetsAcked counter can overflow. An implementation has to deal with tcpEStatsAppHCThruOctetsAcked overflows.

5. References

5.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, DOI 10.17487/RFC4898, May 2007, <https://www.rfc-editor.org/info/rfc4898>.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <https://www.rfc-editor.org/info/rfc5961>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

5.2. Informative References

[SP2024Spoof]
Pan, Y. and C. Rossow, "TCP Spoofing: Reliable Payload Transmission Past the Spoofed TCP Handshake", DOI 10.1109/SP54263.2024.00182, May 2024, <https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00182>.

Acknowledgements

This template uses extracts from templates written by Pekka Savola, Elwyn Davies and Henrik Levkowetz.

We thank Eric Dumazet for proposing the second mitigation option using tcpEStatsAppHCThruOctetsAcked.

Authors' Addresses

Yepeng Pan
CISPA Helmholtz Center for Information Security
Email: yepeng.pan@cispa.de

Christian Rossow
CISPA Helmholtz Center for Information Security
Email: rossow@cispa.de