CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Improve TCP Handling of Out-of-Window Packets to Mitigate Ghost ACKs

Prof. Dr. Christian Rossow and Yepeng Pan
{rossow@cispa.de, yepeng.pan@cispa.de}

[1] Improve TCP Handling of Out-of-Window Packets to Mitigate Ghost ACKs (ietf.org)

List of Content

Overall, the ID focuses on the **ACK number validation** for **established TCP connections** and aims to solve a problem that allows segments with out-of-window ACK numbers to be accepted (**“Ghost ACK”**).

- Current RFC Review and Vulnerability Description
- Proposed RFC Mitigation and Updates

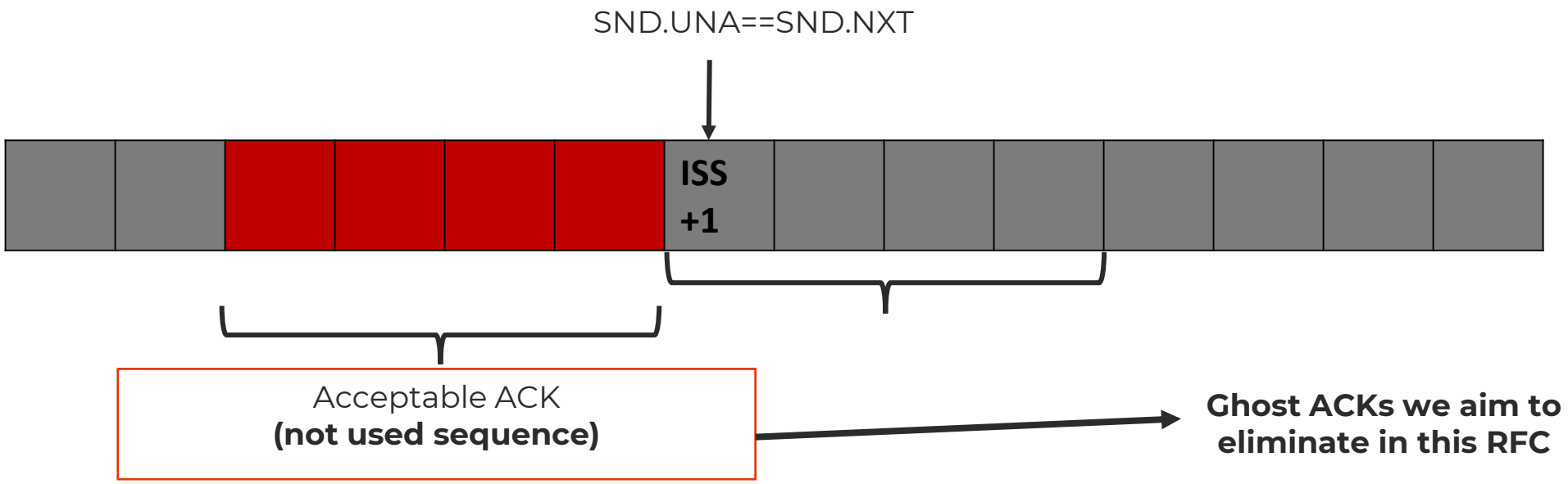
Problem Review

For established TCP connections, the current RFC allows packets with:

!SEG.ACK>SND.NXT

Consequently, a newly established TCP connection may accept packets with:

SEG.ACK < ISS < SND.NXT.



Change Overview

In the last version, our RFC proposed two mitigation options to check the ghost ack problems. Since then, we have incorporated several updates.

1. The updated RFC draft assumes RFC 5961 support
2. We updated the generic mitigation option according to the FreeBSD patch
3. We fixed the statement for RFC4898 specific mitigation option when the **`tcpEStatsAppHCThruOctetsAcked`** counter overflows.

Update 1 – Assume RFC 5961 Support

RFC5961 proposed to apply stricter checks over the ACK value of incoming segments, i.e., instead of !SEG.ACK>SND.NXT, RFC 5961 enforces:

```
((SND.UNA - MAX.SND.WND) <= SEG.ACK <= SND.NXT)
```

Our draft suggests to apply stricter SEG.ACK checks for new TCP connections to avoid SEG.ACK < ISS.

As assuming RFC 5961 support can simplify the draft, and TCP stacks that does not support RFC5961 is unlikely to adopt the draft suggested change. Thus, we decided to assume RFC5961 support in general.

Update 2 – Generic Mitigation Option

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

NO_ISS_CHECK flag is set to false for newly established connections to track if ghost ack can still happen.
When the last acceptable old ACK number is after ISS, ghost ack cannot happen again, NO_ISS_CHECK is set to true.

```
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;
    }
}
```

When Ghost ACK is still possible, check if ISS+1 should be used as the minimal possible ACK value for SEG.ACK checking.

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

For too old ACK values, send a challenge ACK and refrain from further processing.

Update 3 – RFC 4898 specific mitigation update

RFC4898 proposed the **`tcpEStatsAppHCThruOctetsAcked`** statistics, which tracks the number that is already acknowledged by the peer.

After assuming RFC5961 support, the RFC4898 specific mitigation can be simplified to only use the following condition for incoming segments:

```
(SND.UNA - min(MAX.SND.WND, tcpEStatsAppHCThruOctetsAcked)) <= SEG.ACK <= SND.NXT
```

We added the statement that TCP implementations using this mitigation option must deal with the **`tcpEStatsAppHCThruOctetsAcked`** 64-bit counter overflow.

Next Steps

We kindly ask for your suggestions and help to improve and develop the current Internet- Draft.

- Further technical improvements?
- Editorial improvements?