CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

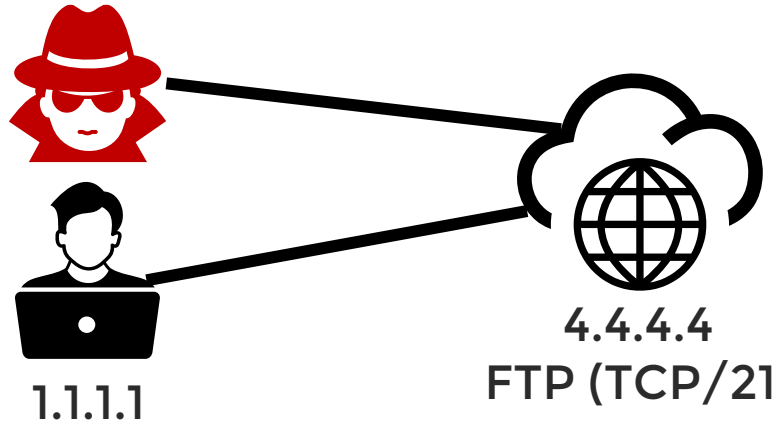
# How Ghost ACKs Enable For Efficient IP-spoofed TCP Connections[1]

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

[1] based on our research paper "TCP Spoofing: Reliable Payload Transmission Past the Spoofed TCP Handshake", IEEE S&P 2024

# TCP Injection attack

- The TCP injection attack enables an off-path attacker to inject a payload into an established connection.
- An attacker has to guess an acceptable SEG.SEQ number and SEG.ACK number.
  - SEG.ACK: As per RFC 9293, segments with SEG.ACK ≤ SND.NXT are acceptable (though SEG.ACK ≤ SND.UNA are duplicate), while segments with SEG.ACK > SND.NXT acknowledge never sent data, and thus are not acceptable.
  - SEG.SEQ: As per RFC 9293, segments with RCV.NXT ≤ SEG.SEQ < RCV.NXT+RCV.WND or RCV.NXT ≤ SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND is judged to occupy a portion of valid receive sequence space.



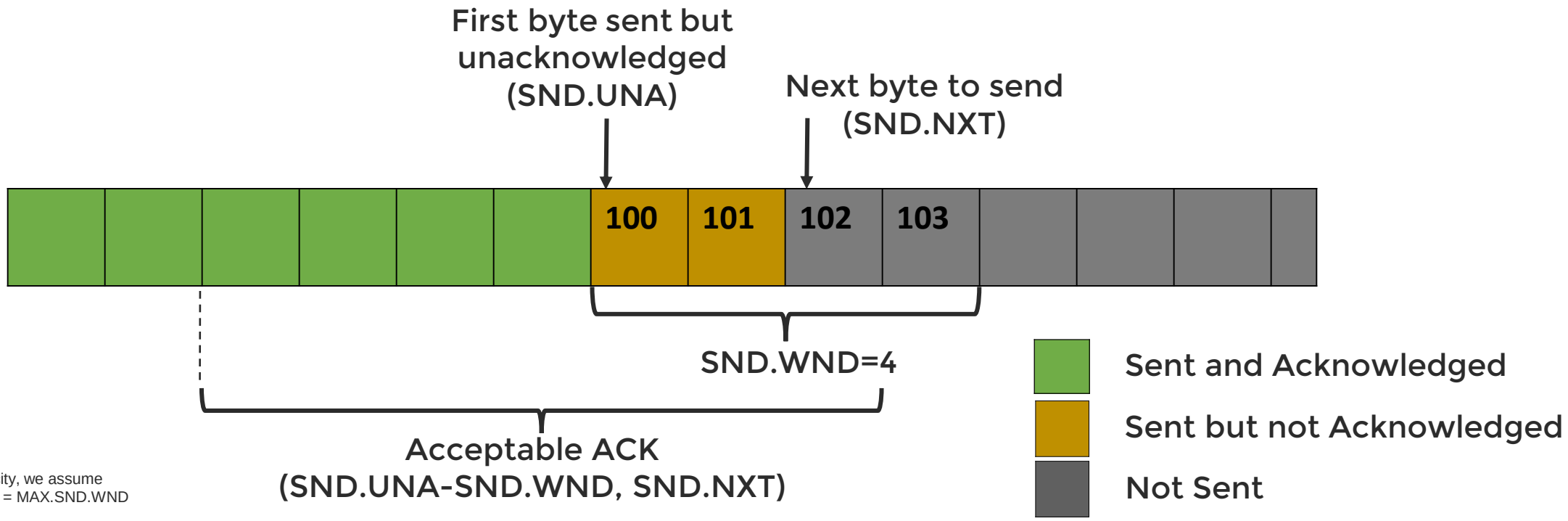

TCP 1.1.1.1:5000→4.4.4.4:21
SEQ=1 “DELE backup.tar”
SEQ=2 “DELE backup.tar”
SEQ=3 “DELE backup.tar”
…

# RFC 5961 specification

- RFC 5961 proposes to apply stricter checks over **acceptable** SEG.ACK numbers:

  The ACK value is considered acceptable only if it is in the range of ((SND.UNA - MAX.SND.WND) <= SEG.ACK <= SND.NXT).

  All incoming segments whose SEG.ACK value doesn't satisfy the above condition MUST be discarded and an SEG.ACK sent back.

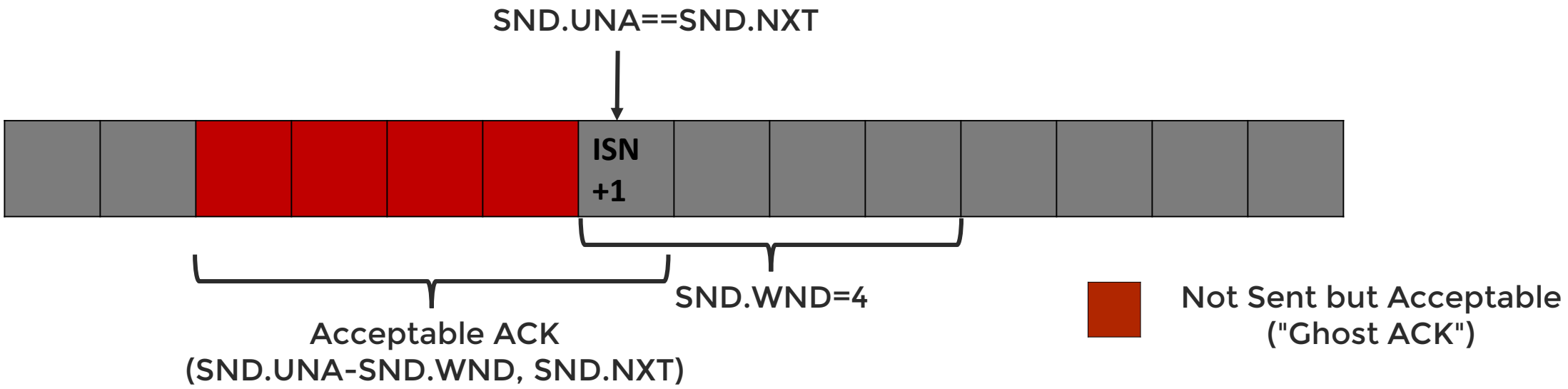





For simplicity, we assume SND.WND = MAX.SND.WND

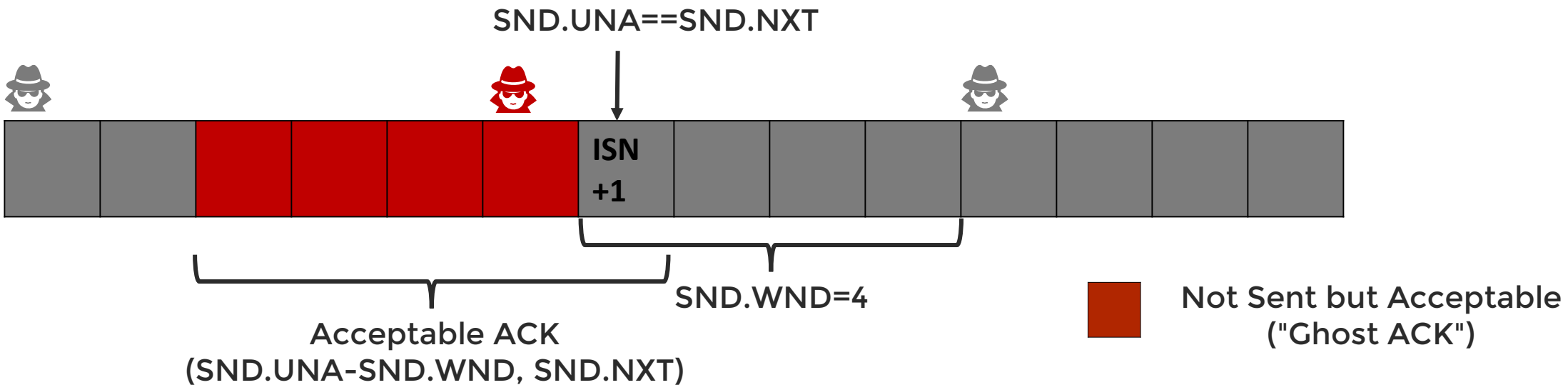
# Ghost ACKs

- The current standards (incl. RFC 5961) do not explicitly treat duplicate ACKs that acknowledge data that was never sent ("Ghost ACKs")
- Standards implicitly interprets Ghost ACKs as “duplicate ACKs“, as they fulfil:
  - **RFC 5961:** `(SND.UNA – MAX.SND.WND) ≤ SEG.ACK ≤ SND.UNA`, and
  - **RFC 9293:** `SEG.ACK ≤ SND.UNA`



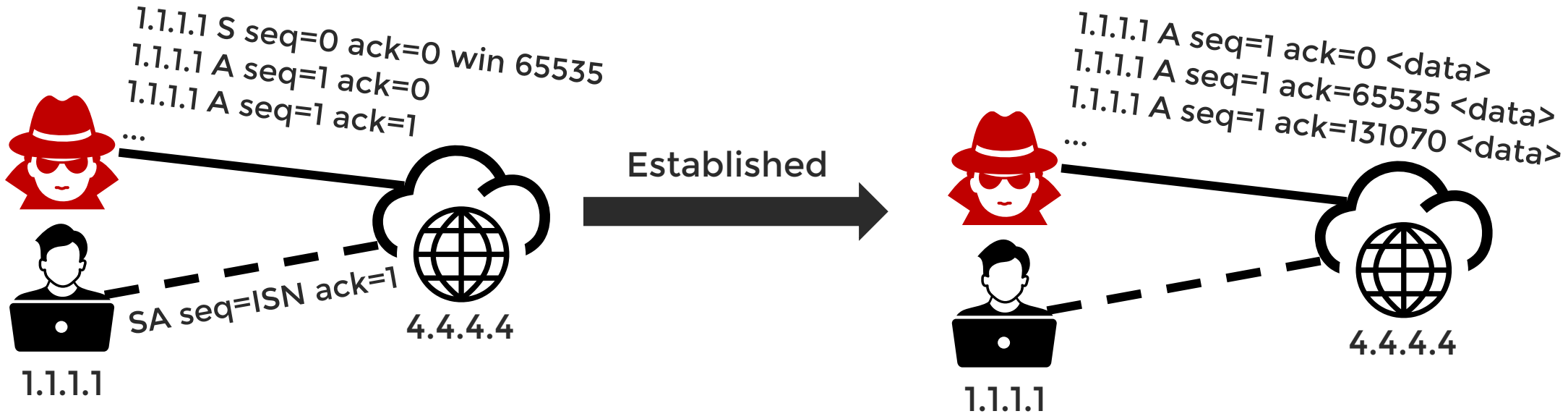
# Security Implications of Ghost ACKs #1: TCP Injection

- The loose SEG.ACK checks ease injection attacks against “new” connections
- Large send windows (e.g., 1 GB) = many acceptable SEG.ACK values (even if not a single byte has been sent yet in that connection)
- Unnecessarily allows attacker to bruteforce correct ACK in new connections → max. $2^{32}$ / SND.WND attempts (e.g., 4x for SND.WND = 1 GB)



# Security Implications of Ghost ACKs #2: TCP Spoofing

- A TCP spoofing attack establishes an IP-spoofed TCP connection to a target via bruteforcing the server-chosen Initial Sequence Number (ISN)
  - Attack motivation: bypass host-based authentication (e.g., SPF, databases, …)
  - TCP spoofing **always** establishes a new connection → Ghost ACKs!
- Ghost ACKs reduce the complexity of injection from ~$2^{32}$ to $2^{32}$ / SND.WND




*Stage 1): Bruteforce spoofed TCP connection*

*Stage 2): Transmit payload into spoofed connection.*

# Prevalence

- Affected TCP/IP stacks
  - We validated the behavior on Windows, Linux, and *BSD.
  - All of these operating systems would accept Ghost ACKs:
    - Packetdrills: https://github.com/ypando/packetdrill_examples

- Authenticated/encrypted connections (TLS, TCP-AO, etc.) are less affected.

# Mitigation

- Linux[3] now mitigates Ghost ACKs by checking the `bytes_acked` (`tcpEStatsAPPHCThruOctetsAcked`) statistics suggested by RFC 4898.
  - `bytes_acked` = number of bytes already acknowledged by sender
  - **SND.UNA – min(MAX.SND.WND, bytes_acked) ≤ SEG.ACK ≤ SND.NXT**
- The above restriction can ensure that for a newly established connection, Linux first verifies if SEG.ACK is within the range of already sent bytes and thus mitigates Ghost ACKs



[2] https://lore.kernel.org/netdev/20231205161841.2702925-1-edumazet@google.com/T/#u

# Summary

- Ghost ACKs = ACKs within send window that acknowledge unsent data
- Ease TCP payload injection, especially for IP-spoofed TCP connections
- Major OSes affected, Linux already patched

- Do we need to address Ghost ACKs in the standards? (And if so, please help.)