Bangladesh University of Engineering and Technology

Department of Computer Science and Engineering

Academic Year 2024–2025

CSE 406

Computer Security Sessional

TCP WindowScaling Attack

Submitted by:

1905082 — Mst. Fahmida Sultana Naznin

1905080 — Mohammad Ninad Mahmud Nobo

Supervised by:

Md. Ashrafur Rahman Khan

Adjunct Lecturer

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Submission Date: July 17, 2025

Contents

1 INTRODUCTION TO TCP WINDOW SCALING ATTACK

1.1 WHAT IS TCP WINDOW SIZE?

The TCP window size represents the amount of data the receiving host can accept before it must send an acknowledgment back to the sender. It acts as a buffer zone that helps prevent overflow on the receiving side and maintains efficient data transmission. Traditionally, this size was limited to 65,535 bytes (64KB), due to the 16-bit field in the TCP header that represents the window size. This restriction was sufficient for legacy networks with limited bandwidth. However, for modern high-speed networks, the limitation poses a performance bottleneck, particularly when the *Bandwidth-Delay Product (BDP)* exceeds the available window size.

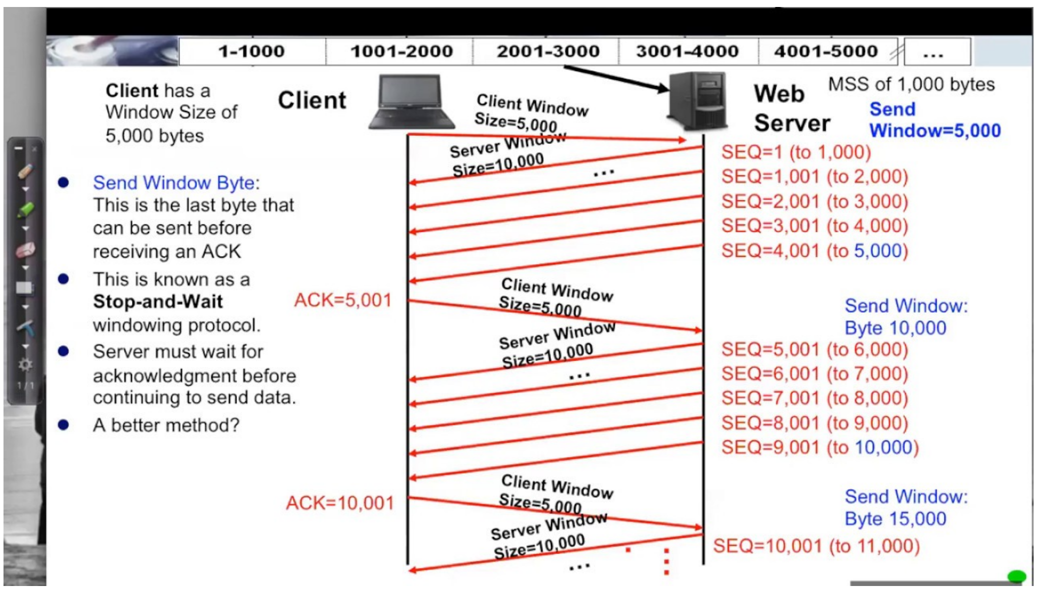

Figure 1: An Overview of TCP Flow Control and Window Size

1.2 WHY TCP WINDOW SCALING IS NEEDED?

To address the 64KB limit, TCP Window Scaling was introduced through the TCP Extensions for High Performance as defined in RFC 1323. This mechanism allows the sender and receiver to negotiate a scaling factor during the initial three-way handshake. By left-shifting the 16-bit window size field, the effective window size can be increased up to 1 Gigabyte (2^{30} bytes), enabling better performance over high-latency, high-bandwidth networks.

The scaling option is activated if both endpoints advertise support for it in their SYN packets. Even if the scaling factor is zero, its presence signifies compatibility. Once established, the window can scale dynamically based on the receiver's ability to process incoming data, thus minimizing idle time and improving throughput.

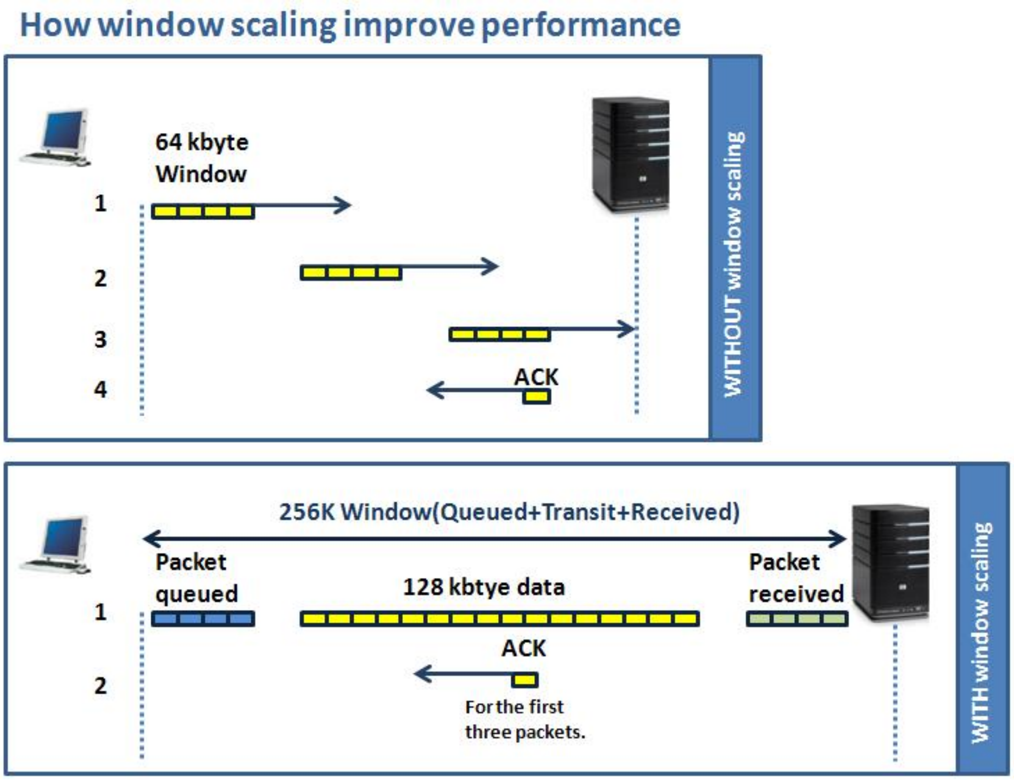

Figure 2: Illustration of TCP Window Scaling Enhancens Throughput

1.3 TCP WINDOW SCALING IN OPERATING SYSTEMS

WINDOWS-BASED SYSTEMS

Modern Microsoft Windows versions (both clients and servers) typically have TCP window scaling enabled by default. Manual tuning is rarely required, but verification can be done using PowerShell or registry inspection tools. In performance-critical environments, administrators may need to inspect or fine-tune these configurations to ensure consistency and optimal performance.

LINUX-BASED SYSTEMS

In Linux, TCP window scaling is not always enabled by default. It can be checked and enabled using the following commands:

```
sysctl net.ipv4.tcp_window_scaling
sudo sysctl -w net.ipv4.tcp_window_scaling=1
```

To make the change persistent, it should be added to the `/etc/sysctl.conf` file:

```
net.ipv4.tcp_window_scaling=1
```

Administrators can further tune send and receive buffers using:

- `net.ipv4.tcp_rmem` – receive buffer (min, default, max)
- `net.ipv4.tcp_wmem` – send buffer (min, default, max)

Proper configuration is critical for achieving full bandwidth utilization on modern networks.

1.4 TCP WINDOW SCALING ATTACK

The TCP Window Scaling Attack exploits the TCP window management mechanism to disrupt data flow between a client and a server. TCP uses the *window size field* to control how much data the sender can transmit before receiving an acknowledgment (ACK). The Window Scale option, negotiated during the TCP three-way handshake (SYN, SYN-ACK, ACK), multiplies the window size to support larger data transfers in high-bandwidth networks.

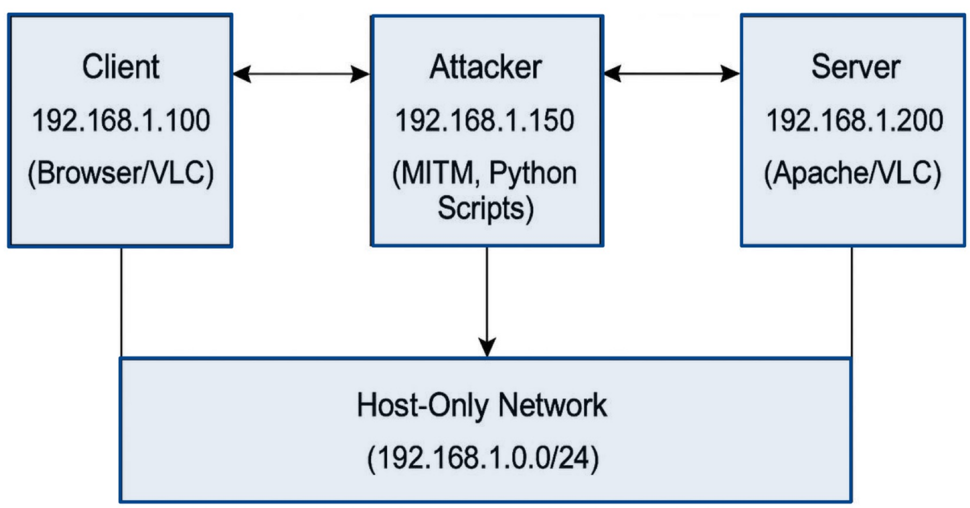

Figure 3: Topology Diagram of TCP Window Scaling Attack

VARIANTS OF TCP WINDOW SCALING ATTACKS

- **Fake Window Size Attack (Denial-of-Service)**
  Attacker sends fake packets with extremely small or large window sizes to disrupt flow or overload receiver.

- **Scaling Factor Trick Attack**
  Attacker modifies the scaling factor during handshake to confuse endpoints and degrade performance.
- **Zero-Window Attack**
  Attacker repeatedly sends packets with window size set to zero, causing sender to pause transmission.
- **Oversized Window Attack**
  Attacker uses a large window size with scaling to flood receiver and cause resource exhaustion.
- **Bypassing Security Systems (IDS Evasion)**
  Attacker uses abnormal or inconsistent window sizes to evade intrusion detection systems.
- **Probing Attack (Information Gathering)**
  Attacker sends crafted packets to learn window size behavior and scaling support for future attacks.

ATTACK IMPACT

- **Window Scale = 0:**
  Forces the Server to send smaller data chunks (maximum of 65,535 bytes). This increases the frequency of ACKs, resulting in slower data transfer rates.
- **Window Size = 0:**
  Causes the Server to pause all data transmission. As a result, the Client's application (e.g., browser or VLC) stalls. This achieves a stealthy Denial of Service (DoS) with minimal risk of detection.

2 OUR PROPOSED ATTACK METHODOLOGY

We propose **Interval-Zero Window Size Attack** which is a stealthy method to disrupt TCP communication by intermittently manipulating the TCP window size field to slow down or pause data transmission between a Client and a Server. This attack leverages TCP's flow control mechanism, where the window size in a TCP header indicates how much data the receiver can accept before sending an acknowledgment (ACK). By setting the window size to 0 in selected data packets (specifically, every 5th ACK packet from the Client to the Server), the Attacker causes the Server to temporarily halt data transmission, mimicking legitimate network congestion. This intermittent approach ensures the attack is harder to detect while still causing significant delays or a partial Denial of Service (DoS).

2.1 ATTACK STRATEGY

1. **Man-in-the-Middle (MITM):**
   Use *ARP poisoning* to intercept network traffic between the Client and the Server.

2. **Packet Sniffing:**
   Capture `TCP ACK` packets from the Client using *raw sockets* or packet capture tools.

3. **Packet Modification:**
   Modify every 5th `ACK` packet by setting the *Window Size* field to 0. Recalculate and update the TCP checksum to ensure packet validity.

4. **Packet Forwarding:**
   Forward both modified and unmodified packets to the Server to maintain seamless connectivity and avoid detection.

Figure 4: Timing Diagram for Our Attack Strategy

3 PACKET DETAILS AND MODIFICATIONS

3.1 TCP PACKET STRUCTURE

The TCP header is typically 20 bytes long in its base form, with optional fields (e.g., Window Scale, Timestamp) extending its length. Each field is crucial for packet manipulation in your attack. The format is as follows:

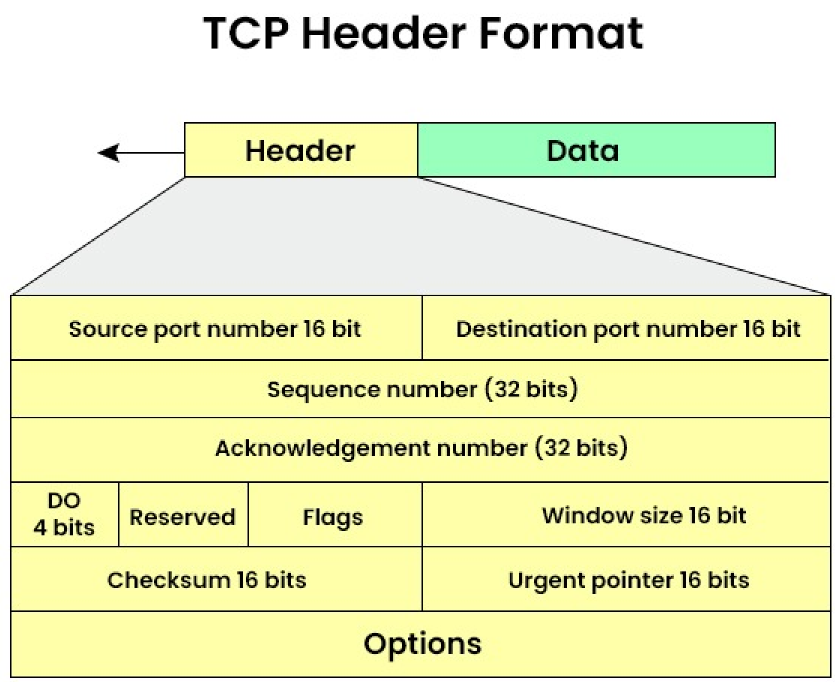

Figure 5: TCP Packet Format

3.2 ATTACK MODIFICATIONS

The attack modifies every 5th ACK packet from Client to Server.

Modified ACK Packet:

- **Original**: Flags = ACK (0x10), Window Size = 65,535 (or scaled, e.g., ˜8.4 MB).
- **Modified**: Flags = ACK (0x10), Window Size = 0.
- **Frame Details**:
  - Ethernet: Source = Attacker's MAC (00:0c:29:zz:zz:zz), Destination = Server's MAC (00:0c:29:yy:yy:yy).

Table 1: TCP Header Format (20 Bytes Base, Additional Bytes for Options)

Offset	Size (Bits)	Field Name	Description
0	16	Source Port	Identifies the sending application (e.g., 80 for HTTP).
2	16	Destination Port	Identifies the receiving application (e.g., 80 for Server).
4	32	Sequence Number	Initial sequence number or byte offset of data (used in ACK packets).
8	32	Acknowledgment Number	Next expected byte from the sender (set in ACK packets).
12	4	Data Offset	Length of TCP header in 32-bit words (e.g., 5 for 20 bytes).
12	3	Reserved	Must be zero (unused).
12	9	Flags	Control bits (e.g., ACK, SYN, FIN; ACK = 0x10).
14	16	Window Size	Receiver's buffer capacity (0–65,535 bytes, modified to 0 in attack).
16	16	Checksum	Ensures data integrity (recalculated after modification).
18	16	Urgent Pointer	Points to urgent data (if URG flag is set, otherwise 0).
20	Variable	Options	Optional fields (e.g., Window Scale, Timestamp; up to 40 bytes).

- IP: Source = 192.168.1.100, Destination = 192.168.1.200.
- TCP: Preserve sequence/acknowledgment numbers, recalculate checksum.

4 JUSTIFICATION: WHY THE DESIGN SHOULD WORK

4.1 WHY IT WORKS

The Interval-Zero Window Size Attack exploits TCP's trust in the window size field (RFC 793). By setting Window Size = 0 in every 5th ACK packet, the Attacker pauses Server transmission, causing Client delays. The intermittent approach ensures stealth by mimicking legitimate flow control.

Table 2: Packet Modifications

Packet Type	Field Modified	Original Value	Modified Value
Data (ACK)	Window Size	e.g., 65,535 (Scaled)	0 (every 5th packet)

Table 3: Justification for Attack Effectiveness

Aspect	Explanation	Why Effective
MITM via ARP Poisoning	Redirects traffic using fake ARP replies.	Local networks are vulnerable; VirtualBox supports promiscuous mode.
Zero-Window Attack	Sets Window Size = 0 in 5th ACK packets.	Ubuntu's TCP stack halts on Window Size = 0, causing delays.
Intermittent Strategy	Modifies every 5th ACK.	Mimics congestion, evading basic detection.
Checksum Recalculation	Ensures valid TCP checksums.	Prevents packet rejection, maintaining transparency.
Stealth	Zero-window mimics legitimate behavior.	Hard to detect without deep packet inspection.

4.2 INTERVAL-ZERO SPECIFICS

- **Why Chosen**: Intermittent zero-window attacks maximize impact (delays/partial DoS) while minimizing detection risk by resembling congestion.
- **Impact**:
  - **Client**: Browser stalls (e.g., "Loading" every ˜2 MB of a 10 MB file); VLC buffers.
  - **Server**: Pauses for ˜1–2 seconds per zero-window packet.
  - Example: 10 MB download pauses 3–5 times, taking ˜30–60 seconds.
- **Detection Difficulty**:
  - **Server**: Sees Window Size = 0 as a legitimate full buffer signal.
  - **Client**: Experiences stalls without explicit errors.
  - **Monitoring**: Requires Wireshark (filter: `tcp.window_size == 0`) to detect spoofed packets.
  - **Interval**: Every 5th ACK avoids continuous stalls, blending with normal traffic.
- **Lab Feasibility**: VirtualBox ensures packet capture; Ubuntu TCP stack respects zero-window states.

5 CONCLUSION

The Interval-Zero Window Size Attack disrupts TCP communication by intermittently setting Window Size = 0 in ACK packets, causing sporadic Server pauses. Its stealthy design, mimicking legitimate flow control, ensures effectiveness and low detectability in a VirtualBox lab. The attack leverages ARP poisoning and raw socket programming, with Wireshark verification, making it ideal for demonstrating impact for Project 2025.