

Q1. Server was started on port 5123.

Screenshot1

The screenshot displays three windows. The top window is Wireshark, showing a packet capture on the loopback interface. The packet list shows several TCP packets between 127.0.0.1 and 127.0.0.1. The packet details pane shows the selected packet (No. 1) with its structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The bottom two windows are PowerShell. The left window shows the execution of `java TwoWayAsyncMsgServer 5123`, which outputs 'Waiting for a client ...', 'Connected to a client at ('127.0.0.1', '50728')', 'Client: Nathan Bickel', 'That's me', 'Client: What's your username?', 'nbickel', 'bye', and a connection reset exception. The right window shows the execution of `java TwoWayAsyncMsgClient loopback 5123`, which outputs 'Connected to server at loopback:5123', 'Nathan Bickel', 'Server: That's me', 'What's your username?', 'Server: nbickel', 'Server: bye', '^Z', and '*** Client closing connection'.

Screenshot 2

The screenshot shows the Wireshark interface with a display filter of `tcp.port == 5123`. The packet list shows several TCP packets between 127.0.0.1 and 127.0.0.1. The packet details pane shows the selected packet (No. 1037) with its structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data of the selected packet.

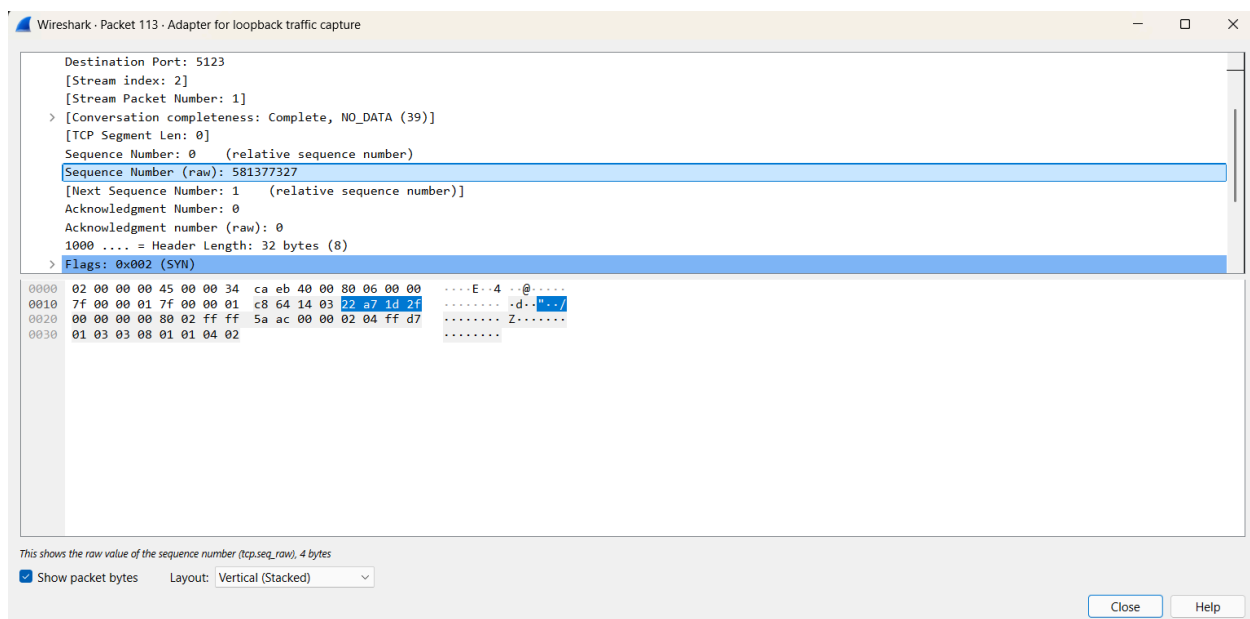
Q2. Client's TCP Source port is 50728.

Screenshot 3

The screenshot shows the Wireshark interface with a display filter of `tcp.port == 5123`. The packet list shows several TCP packets between 127.0.0.1 and 127.0.0.1. The packet details pane shows the selected packet (No. 1092) with its structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data of the selected packet.

Q3. TCP Sequence numbers don't always start at 0 to resolve issues with duplicate connections and security. Consider if sequence numbers did always start at 0. Then, if a connection were opened and closed twice in quick succession, sequence numbers from the first connection may linger and cause the second connection to believe sequence numbers are being duplicated, causing the second connection to start resending packets when doing so is not necessary. By starting the sequence at a different random number each time the connection is reopened, this overlap will (almost certainly) not be an issue. Having randomized starting sequence numbers also makes it harder for malicious actors to predict future sequence numbers and thus doing so acts as a safeguard against attacks.

Screenshot 4



Q4. The Window Scale Factor is used to allow TCP to support larger receive windows, which is important for high-speed or long-distance networks. Without it, the maximum window size is limited to 65,535 bytes, which can become a bottleneck when trying to transfer a lot of data quickly. For example, on a fast connection with high latency, a small window would often require the sender to wait for acknowledgments before continuing, slowing things down. The Window Scale Factor solves this by letting the receiver tell the sender to interpret the window size as multiplied by the specified factor. This allows the sender to send more data without waiting and improves performance on networks where large amounts of data are expected.

Screenshot 5

Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	45	49678 → 49679 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=1
2	0.000034	127.0.0.1	127.0.0.1	TCP	44	49679 → 49678 [ACK] Seq=1 Ack=2 Win=164 Len=0
3	0.000700	127.0.0.1	127.0.0.1	TCP	45	49681 → 49680 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=1
4	0.000726	127.0.0.1	127.0.0.1	TCP	44	49680 → 49681 [ACK] Seq=1 Ack=2 Win=164 Len=0
5	0.001255	127.0.0.1	127.0.0.1	TCP	45	49681 → 49680 [PSH, ACK] Seq=2 Ack=1 Win=255 Len=1
6	0.001271	127.0.0.1	127.0.0.1	TCP	44	49680 → 49681 [ACK] Seq=1 Ack=3 Win=164 Len=0
7	0.001682	127.0.0.1	127.0.0.1	TCP	45	49678 → 49679 [PSH, ACK] Seq=2 Ack=1 Win=255 Len=1
8	0.001695	127.0.0.1	127.0.0.1	TCP	44	49679 → 49678 [ACK] Seq=1 Ack=3 Win=164 Len=0
9	0.002128	127.0.0.1	127.0.0.1	TCP	45	49681 → 49680 [PSH, ACK] Seq=3 Ack=1 Win=255 Len=1
10	0.002148	127.0.0.1	127.0.0.1	TCP	44	49680 → 49681 [ACK] Seq=1 Ack=4 Win=164 Len=0
11	0.002571	127.0.0.1	127.0.0.1	TCP	45	49678 → 49679 [PSH, ACK] Seq=3 Ack=1 Win=255 Len=1
12	0.002583	127.0.0.1	127.0.0.1	TCP	44	49679 → 49678 [ACK] Seq=1 Ack=4 Win=164 Len=0
13	0.007179	127.0.0.1	127.0.0.1	TCP	45	49681 → 49680 [PSH, ACK] Seq=4 Ack=1 Win=255 Len=1
14	0.007206	127.0.0.1	127.0.0.1	TCP	44	49680 → 49681 [ACK] Seq=1 Ack=5 Win=164 Len=0
15	0.009046	127.0.0.1	127.0.0.1	TCP	45	49678 → 49679 [PSH, ACK] Seq=4 Ack=1 Win=255 Len=1
16	0.009065	127.0.0.1	127.0.0.1	TCP	44	49679 → 49678 [ACK] Seq=1 Ack=5 Win=164 Len=0
17	0.009497	127.0.0.1	127.0.0.1	TCP	45	49681 → 49680 [PSH, ACK] Seq=5 Ack=1 Win=255 Len=1
18	0.009521	127.0.0.1	127.0.0.1	TCP	44	49680 → 49681 [ACK] Seq=1 Ack=6 Win=164 Len=0

[TCP Segment Len: 1]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 4174731855
[Next Sequence Number: 2 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3993453200
0101 = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
Window: 255
[Calculated window size: 255]
[Window size scaling factor: -1 (unknown)]
Checksum: 0xa7ee [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]
TCP payload (1 byte)
Data (1 byte)

0000 02 00 00 00 45 00 00 29 45 ab 40 00 80 06 00 00E... E @....
0010 7f 00 00 01 7f 00 00 01 c2 0e c2 0f f8 d5 5a 4fZ0
0020 ee 07 42 90 50 18 00 71 a7 ee 00 00 01 ...B-P... ..

The window size scaling factor (-1 when unknown, -2 when no scaling is used) (tcp.window_size_scalefactor), 2 bytes

Packets: 36 - Dropped: 0 (0.0%)

Profile: Default