On Accurate Measurements of Bytes Transmitted in Network Sessions

Todd Heberlein

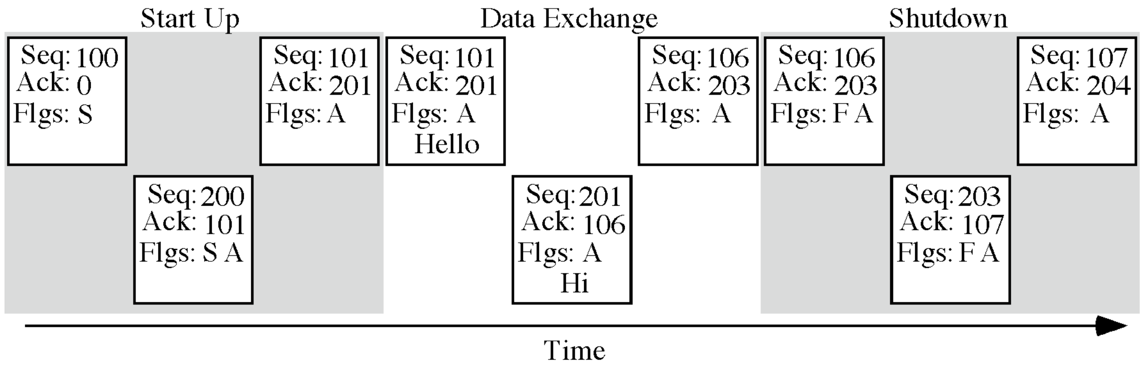
1 Introduction

During the first in a series of experiments to identify potential statistical relationships between network session observables for attacks and target types (e.g., vulnerable servers vs. patched servers), we identified a flaw in our use of TCP sequence numbers as a means to measure the number of bytes transmitted by a host. As organizations evolve from using simple signature-based sensors to exploiting information contained within network flow summaries, accurate representation of those flows will become more important. For this reason, we have written this report to document the flaw in our first round of experiments and to explain the subtle interaction of the TCP sequence numbers and TCP control flags so that others can avoid our mistakes.

Section 2 summarizes the interaction between TCP sequence numbers and TCP control flags. Section 3 presents the relevant portions of our network flow summary, and Section 4 shows how our approach prevented us from arriving at an accurate measure of the number of bytes transmitted by a host. Finally, Section 5 summarizes our findings and touches on our future plans.

2 TCP Sequence Numbers and Flags Interaction

To summarize the relevant portions of a network flow, we will use an imaginary application called ShortWord. A client connects to the ShortWord server, sends a word, and the server responds with a shorter word with similar meaning (or the original word if no such word can be found). The connection will remain open until the client chooses to terminate the connection.


Start Up
Data Exchange
Shutdown
Seq: 100 Ack: 0 Flgs: S
Seq: 200 Ack: 101 Flgs: S A
Seq: 101 Ack: 201 Flgs: A
Seq: 101 Ack: 201 Flgs: A Hello
Seq: 201 Ack: 106 Flgs: A Hi
Seq: 106 Ack: 203 Flgs: A
Seq: 106 Ack: 203 Flgs: F A
Seq: 203 Ack: 107 Flgs: F A
Seq: 107 Ack: 204 Flgs: A
Time

Sequence	100	101	102	103	104	105	106	107
Data		H	e	l	l	o		

Client Sequence Numbers

Sequence	200	201	202	203
Data		H	i	

Server Sequence Numbers

Figure 1: Network Flow and Sequence Numbers

Figure 1 shows a simplified network flow for the ShortWord application. At the top is a flow of packets from left to right. The top row contains packets from the client, and the bottom row represents packets from the server. Each packet shows sequence and acknowledgement numbers (labeled "Seq" and "Ack"), TCP control flags (labeled "Flgs"), and data if any exists. Visually, we divided the packet flow into three groups: a three-way handshake to establish the connection, a period when data is exchanged, and three packets for session shutdown. The number of packets for start up should always be three, but the number of packets for data exchange and session shutdown may vary. In addition to a clean shutdown with TCP FIN flags, one or both hosts can send a reset (RST) packet to immediately terminate the connection.

Below the packet flow are two tables representing the sequence numbers observed and data from each host. For example, the letter 'e' in "Hello" from the client is associated with sequence number 102. The numbers are directly tied to the sequence numbers shown in the packet flow shown above the tables.

During the connection start up phase the client sends a SYN packet with an initial sequence number (ISN) of 100. The server responds with a SYN-ACK packet with an ISN of 200 and an acknowledgment of 101. Finally the client sends an ACK packet with sequence number of 101 (one more than its ISN, and the same number the server sent back in its acknowledgement field) and an acknowledgement of 201 – one more than the server's ISN. The initial sequence numbers are not associated with any data; the first byte from each host is associated with a sequence number that is one more than the ISN. From the perspective of the sequence and acknowledgement numbers, a SYN flag is, in effect, treated like a byte of data.

During the data exchange phase, each byte is associated with a sequence number; there is a one-to-one mapping. The acknowledgement number lets the other host know which bytes have been successfully received. If one host (e.g., the server) does not acknowledge some data sent within a given window of time, the other host (e.g., the client) will resend the unacknowledged data. There are many reasons that one host may not have acknowledged data including: (1) the data packet was dropped in transmission, (2) the data packet arrived at the destination but the data was corrupted in transmission, (3) the acknowledgement packet was dropped, (4) the acknowledgement arrived but the packet was corrupted, or (5) the acknowledgement packet will eventually arrive but not in a timely manner.

During a clean shutdown one of the hosts sends a FIN-ACK packet signaling that it has no more data to send. If the second host also has no more data to send, it will also send a FIN-ACK packet and increment the acknowledgement number even though *it received no data*. Finally, the host that initiated the shutdown will send a final ACK packet with incremented sequence and acknowledgement numbers. As in the SYN packet cases, the FIN flag is treated like a data byte and the appropriate sequence and acknowledgement numbers are incremented.

3 Our First Approach

For our first experiment we captured several variables related to a host's activity for a single session. Table 1 shows the relevant variables for this discussion and some sample values from a session. The "payload bytes" field is the sum of TCP payload bytes for all packets sent by the host. Bytes retransmitted will be counted multiple times. The "packets" field is the total number of packets sent by the host. The "first sequence" number is the first TCP sequence number observed from this host; if this sequence number came from a SYN or SYN-ACK packet, this value will be the ISN. The "last sequence" field is the last sequence number in a packet from the host. This "first flags" field is the TCP control flags for the first packet sent by the host. The "composite flags" field indicates all the TCP control flags used by the host during the session; it is essentially a large "logical or" of all flags for all packets from the host.

From the example in Table 1 we can determine the host initiated the connection (because of the lone SYN flag in the "first flags" field) and that 6 packets were observed carrying a total of

240 bytes of data. Subtracting the first sequence number from the last sequence number gives us a value of 242, a value that closely corresponds to the total payload bytes observed. Since the host sent packets with both SYN and FIN flags, we can treat these as pseudo bytes, giving us a total of 242 payload bytes (real data bytes plus the pseudo bytes), a number that agrees with the difference between the first and last observed sequence numbers.

Table 1: Relevant Flow Information For One Host

Variables	Payload Bytes	Packets	First Sequence	Last Sequence	First Flags	Composite Flags
Example	240	6	837987945	837988187	____S_	_AP_SF

4 The Problem

In the example shown in Table 1, there appeared to be no reransmission or loss of payload data (or the retransmissions and losses exactly canceled each other out), so the "payload bytes" field accurately reflected the number of bytes sent by the host. However, in general this is not the case, and the subtraction of the first sequence number from the last sequence number generally provides a more accurate calculation of the number of bytes sent. The problem is that by only observing sequence numbers, our analysis can be off by one byte of data.

The bottom tables for the client and server sequence numbers in Figure 1 illustrates the problem. The difference between the TCP sequence numbers observed from the client is 7 (107-100), while 5 bytes of actual data were transmitted – a difference of 2. Meanwhile, the difference between the TCP sequence numbers observed from the server is 3 (203-200), while 2 bytes of actual data were transmitted – a difference of only 1. In short, by only using observed sequence numbers we are unable to determine the *exact* number of bytes transmitted by the host because we do not know whether the difference between the sequence number range and actual bytes transmitted is 1 or 2.

The root of the problem is that our sensor did not take in to account that the FIN flag effectively increases the sequence number. In one case (the client in our example) the host sends another packet after the FIN packet with the larger sequence number, while in the other case (the server in our example), no subsequent packets are sent after the FIN packet.

In general, if the host initiated the connection shutdown the difference between the sequence numbers and the actual number of bytes transmitted will be 2; otherwise, the difference will be 1. Either the client or the server can initiate a shutdown, so we cannot predict from which host we should subtract 2 and which we should subtract 1 from the sequence number delta.

5 Conclusions

Because of subtle interactions between TCP sequence numbers and TCP control flags, determining the exact number of bytes transmitted by a host using only observed sequence numbers is impossible. While future experiments with a modified sensor can correct this problem, with the data collected for our first round of experiments we cannot reliably correct for this error. Furthermore, the analysis of the underlying cause of the error we were observing has led us to a new appreciation of knowing which host initiated a shutdown, and we wonder if this knowledge could, in some circumstances, be a reliable predictor for the effects (e.g., success or failure) of an attempted attack. In addition to fixing the sensor to account for the observed sequence number problem, we shall also record which host initiated session shutdown.