

Studying Normal Traffic, Part Three: TCP Headers

Karen Kent Frederick 2001-05-14

Studying Normal Traffic, Part Three: TCP Headers

by *Karen Kent Frederick*

last updated May 14, 2001

This is the final article in a three-part series devoted to studying normal traffic. As was explained in [Studying Normal Traffic, Part One](#) and [Studying Normal Traffic, Part Two: Studying FTP Traffic](#), we often focus our attention on the characteristics of suspicious packets without first becoming familiar with the characteristics of normal traffic. A good and easy way of doing this is to generate, capture and examine your own normal traffic. The first two articles in this series showed how to capture packets using WinDump and reviewed some of the basics of normal TCP/IP traffic. In this article, we will be looking at two other aspects of normal TCP traffic: the structure of TCP packets, and the use of TCP options. Note that in order to understand this material, you should already know the fundamentals of TCP/IP.

TCP Packet Structure

If you've read the previous articles in this series or worked with tcpdump or WinDump, you are probably familiar with the format of their output for TCP traffic. If not, here's a quick review of the meaning of the fields:

```
13:10:30.134481 [timestamp] my.win98.box.2172 [source address and port] > anon.ftp.box.21: [destination address and port] S [TCP flags] 71870464:71870464 [sequence numbers] (0) [bytes of data] win 65535 [TCP window size] [TCP options] (DF) [don't fragment flag is set] (ttl 128, id 44644) [time to live value, and IP identification number]
```

Let's start by looking at an example that shows the first packet (SYN) of a TCP three-way handshake. It was initiated by a Windows 98 computer and the destination machine is an anonymous FTP server.

```
13:10:30.134481 my.win98.box.2172 > anon.ftp.box.21: S
71870464:71870464(0) win 65535 (DF) (ttl 128, id 44644)
```

Most of the fields shown in this entry have been discussed in the previous articles in this series.

One field that we haven't discussed at all are the TCP options; in this example, they are: . So what do they mean, and are they normal or not?

Before we can answer that, we need to know more about the structure of TCP packets themselves. The log entry shown above is actually a text representation of the values of some of the bits and bytes in the packet; the log snippet below shows the actual values of the entire packet in hex. Each pair of numbers (such as "45") constitutes a byte. You can view packets in this way by running tcpdump or WinDump with the -x option.

```
4500 0030 ae17 4000 8006 f398 xxxx xxxx
yyyy yyyy 0c37 0015 0526 a98f 0000 0000
7002 ffff 6f06 0000 0204 05b4 0101 0402
```

The first part of the packet, shown here in italics, is the IP header. It contains several values, including the source and destination IP addresses, the IP identification number, the type of service (TOS), fragmentation information, and the time to live (TTL). The basic IP header is 20 bytes long; if IP options were present, they would add additional bytes to the header. Examples of IP options include loose source routing, strict source routing, timestamps and record route.

Because the use of IP options changes the length of the IP header, a value is set in the header to indicate how many bytes long the header is. It's important to note that the length of the IP header must be a multiple of 4 bytes. In the example shown above, the second hex digit is 5; this indicates that the length of the IP header is five 4-byte chunks, or 20 bytes total. Since this is the minimum length of the IP header, this tells us that no IP options have been set.

An IP packet is composed of the IP header (with or without options) and the data. In this case, the data section contains TCP; in the above example, the bolded values are the regular portion of a TCP header, which is 20 bytes long without options. The TCP header contains various fields including the source and destination ports, sequence and acknowledgment numbers, window size, TCP flags, urgent pointer, and reserved bits.

Like the IP header, the TCP header may also contain options. And like the IP header, the TCP header's length must be a multiple of 4 bytes. The final 8 bytes in the example, shown in plain text, are the TCP options. (Note that TCP options and IP options are two different things.) Because the TCP options change the length of the TCP header, the length is set in the header. In this case, the value of the first half of the 13th byte - in this case, 7 - means that the TCP header is made of seven 4-byte chunks, or a total of 28 bytes: 20 bytes of standard TCP header

and 8 bytes of TCP options.

Because only one hex value is set aside for indicating the TCP header length, the maximum value that can be set is hex F, which is decimal 15. That means that fifteen 4-byte chunks, or 60 bytes, is the maximum length for the entire TCP header, including TCP options.

TCP Options

Let's take a look at some commonly used TCP options:

Maximum Segment Size (MSS): The Maximum Segment Size (MSS) was defined in [RFC 793](#); it is used to exchange Maximum Transfer Unit (MTU) / Maximum Receive Unit (MRU) sizes during the TCP three-way handshake. Basically, both ends of the connection will state the maximum IP datagram size that they can handle without using fragmentation, and the lower of the two values will be used by both ends. If the MSS option is omitted by one or both ends of the connection, the value 536 bytes will be used (per [RFC 1122](#)). The MSS is only negotiated during the SYN and SYN/ACK packets in the TCP three-way handshake, so the MSS option should only be seen in those packets.

Window Scale (WSCALE): Don't confuse the TCP option called Window Scale with the standard TCP header field called Window Size. The window size is used to record how many bytes of buffer space the host has available for receiving data. Because the window size field is only 16 bits long, this limits the maximum window size to 65,535 bytes. At one time, this was sufficient, but it no longer is. In [RFC 1072](#), the Window Scale (WSCALE) option was defined; it was redefined in [RFC 1323](#). The Window Scale value is used to shift the window size field's value up to a maximum value of approximately a gigabyte. Like the MSS option, the WSCALE option should only appear in SYN and SYN/ACK packets during the handshake. However, if both ends of the connection do not use the WSCALE option, the window sizes will remain unchanged.

Timestamp: The Timestamp option was first defined in its current form in [RFC 1323](#). Its purpose is to track the round-trip delivery time for data in order to identify changes in latency that may require acknowledgment timer adjustments. Timestamps can only be used if both ends of the connection agree to use them, as you'll see shortly. Unlike the MSS and WSCALE options, Timestamp options are typically used throughout a TCP session, so if they are being used, you should see them in most of the packets.

The Timestamp option has two timestamp fields: the Timestamp Value and the Timestamp Echo Reply. When a host first timestamps a packet in a connection, it puts the timestamp in the Timestamp Value field and leaves the Timestamp Echo Reply field set to 0 (generally). When the second host receives that packet and prepares to acknowledge it, it transfers the timestamp from the old packet's Timestamp Value field to the new packet's Timestamp Echo Reply field, and it puts a new timestamp in the Timestamp Value field. So the Timestamp Value field always contains the latest timestamp, while the Timestamp Echo Reply field contains the previous timestamp.

No Operation (NOP): The No Operation (NOP) TCP option, defined in [RFC 793](#), is used to provide padding around other options. The length of the TCP header must be a multiple of 4 bytes; however, most of the options are not 4 bytes long. If the total length of the options is not a 4-byte multiple, one or more 1-byte NOPs will be added to the options in order to adjust the overall length. For example, if there were 6 bytes of options, two NOPs would be added. NOPs are sometimes used between options, particularly if an option needs to start on a certain byte boundary, so it is not unusual to see several NOPs throughout a set of TCP options.

Selective Acknowledgments: The concept of selective acknowledgments was defined in [RFC 1072](#) and redefined in [RFC 2018](#). Normally, when a host acknowledges data, it can only acknowledge the packets up to and including the sequence number immediately before a missing packet. This means that if a thousand packets are received but the second one is missing, the host can only acknowledge the receipt of the first packet, so the sender would have to resend all packets from number 2 through 1000. By using selective acknowledgments, the receiver could acknowledge the receipt of the packets from 3 through 1000, so the sender would only have to resend packet 2.

There are two TCP options for selective acknowledgments:

Selective Acknowledgment Permitted (SackOK): This option simply says that selective acknowledgments are permitted for this connection. SackOK must be included in the TCP options in both the SYN and SYN/ACK packets during the TCP three-way handshake, or it cannot be used. SackOK should not appear in any other packets.

Selective Acknowledgment Data: This option contains the actual acknowledgment information for a selective acknowledgment. It lists one or more pairs of sequence numbers, which define ranges of packets that are being acknowledged.

Analyzing Examples of TCP Options

Now we'll return to the question that I asked at the beginning of the article. The TCP options in the example, a SYN packet from a Windows 98 system, are . What does that mean, and is it normal or not?

Here are the number of bytes that each of the options described above uses in a packet:

- Maximum Segment Size (MSS): 4 bytes
- Window Scale (WSCALE): 3 bytes
- Timestamp (TS): 10 bytes
- No Operation (NOP): 1 byte
- Selective Acknowledgment Permitted (SackOK): 2 bytes
- Selective Acknowledgment Data: 10 bytes (plus 8 bytes for each additional pair of sequence numbers)

Putting together all the items we've discussed, the options appear to be normal:

- specifies a Maximum Segment size of 1460 bytes; it uses 4 bytes of the options. It is appropriate to see MSS used in a SYN packet.
- is filler to provide padding between the mss 1460 and sackOK options. Each NOP takes up 1 byte of the header.
- specifies that selective acknowledgments are permitted. This takes up 2 bytes of the header, and it is also appropriate to use in a SYN packet.

Here's another example from a SYN packet sent by an OpenBSD 2.7 box: . Notice that it starts out with the same four options as the previous example (mss 1460,nop,nop,sackOK) but then adds several more. Here is the meaning of each of the additional fields, which total 16 bytes:

- provides 1 byte of filler.
- says that the Window Scale option is supported by the operating system but that it does not wish to use it. This takes up 3 bytes.
- provides 2 bytes of filler.
- is an initial timestamp. The Timestamp Value field is set to 50473, which is the time that the packet was sent. Since this is the first packet, the Timestamp Echo Reply field is set to 0, as there is no previous timestamp to place there. The timestamp option takes 10

bytes.

Another example of a SYN packet comes from a RedHat Linux 7.0 system: . Compare this example to the previous one; you'll notice that both list the same 4 options: mss, sackOK, timestamp and wscale. But they list them in a different order, and they use nop instructions quite differently. Yet both are examples of normal traffic. The RFCs do not state what order the TCP options should be placed in, so each operating system is free to arrange them in whatever order they please.

The final example comes from a FreeBSD 4.0 box: for a SYN packet, the only option that it supplied was . This is quite different from the earlier examples, but it is perfectly acceptable; it just does not use the same options by default as the other operating systems did.

Conclusion

We have examined the normal use of TCP options. Various operating systems not only use different TCP options by default, but they list the options in different sequences, and they also use NOP's for padding in different ways. This is not abnormal behavior, as long as the length of the options totals a multiple of 4 bytes, and the options make sense. For example, options such as MSS and Window Scale shouldn't be set more than once in a packet, nor should they appear in packets that don't have the SYN flag set.

Because most operating systems use a unique default TCP option combination, TCP options can be used to do "passive fingerprinting" of the operating system of a host that sends you a SYN packet. Also, systems can be fingerprinted based on what options they return in response to a set of options sent from another system. For example, if I send a SYN packet with the SackOK option set, and your system sends a SYN/ACK response without the SackOK option set, I know that your operating system does not support selective acknowledgments. By sending a SYN packet with many different options set and examining the options listed in the SYN/ACK response, I may be able to identify the operating system of the other host.

For more information on TCP options, two great references are TCP/IP Illustrated, Volume 1, by W. Richard Stevens, and Internet Core Protocols, by Eric A. Hall. More information on passive fingerprinting in general is available in [Know Your Enemy: Passive Fingerprinting](#) by Lance Spitzner, and information on fingerprinting using TCP options is available from [Remote OS Detection Via TCP/IP Stack Fingerprinting](#) by Fyodor.

Much thanks to Toby Miller for his suggestion to examine default TCP options.

[Karen Kent Frederick](#) is a senior security engineer for the Rapid Response Team at NFR Security. Karen has a B.S. in Computer Science and is currently completing her Master's thesis in intrusion detection through the University of Idaho. She holds several certifications, including Microsoft Certified Systems Engineer + Internet, Check Point Certified Security Administrator, SANS GIAC Certified Intrusion Analyst, GIAC Certified Unix Security Administrator, and GIAC Certified Incident Handler. Karen is one of the authors and editors of "Intrusion Signatures and Analysis", a new book on intrusion detection that was published in January 2001.

Relevant Links

[Studying Normal Traffic, Part One](#)

Karen Kent Frederick, SecurityFocus

[Studying Normal Traffic, Part Two: Studying FTP Traffic](#)

Karen Kent Frederick, SecurityFocus

[Abnormal Traffic](#)

Karen Kent Frederick, SecurityFocus

[tcpdump](#)

Tcpdump Group

[WinDump](#)

NetGroup

[Privacy Statement](#)

Copyright 2006, SecurityFocus