July 2019
Geoff Huston

TCP MSS Values

At the recent IEPG meeting in Montreal in July 2019 Joel Jaeggli of Fastly talked on the topic of the settings of the TCP Maximum Segment Size (MSS) field in hosts (http://www.iepg.org/2019-07-21-ietf105/joel-cve-2019-11477.pdf). There has been a recent vulnerability published (described in CVE-2019-11477, 11478 and 11479) relating to the Linux TCP implementation and one of the effective workarounds to avoid this problem is to block all TCP connection attempts that use a MSS value of 500 or lower. Joel was motivated to look at what was being sent as MSS values to understand the potential impact of this proposed workaround. I'm also curious about this, and in our data set we have a large collection of recorded TCP handshakes. So let's look at what we see about MSS settings.

The TCP MSS Parameter

The MSS parameter is a part of the options filed in the TCP initial handshake that specifies the largest amount of data that a TCP speaker can receive in a single TCP segment. Each direction of TCP traffic uses its own MSS value, as this is a receiver-specified value. The two ends don't have to agree on a common value, as it acts as a constrain on the sender to send TCP segments no larger than this MSS value, but of course smaller TCP segments can always be sent. This MSS value can vary between the forward and reverse directions of a TCP data flow.

The MSS value does not count the TCP header or the IP header. The received IP datagram containing a TCP segment may be self-contained within a single packet, or it may be reconstructed from several fragmented pieces.

As IP packet fragmentation is an IP level issue, TCP should not directly concern itself with IP fragmentation in any case. In theory. In practice, a judicious setting of TCP MSS sizes that attempts to avoid sending TCP packets that incur IP level packet fragmentation should be avoided!

Conventionally, the MS value for a connection is set by the platform rather than the application, and the setting is applied to all TCP connections, but many operating system platforms provide a hook in the connection API for an application to specify the MSS value for a connection (such as the TCP_MAXSEG socket option).

What is a 'good' MSS Value?

Getting the MSS value "just right" is important.

Too high a value can lead to inefficient TCP sessions and even wedged TCP sessions due to issues with mishandling of IP fragmentation. The problem is that the sender may perform TCP segmentation by using the received MSS value as its guide and the resultant TCP packet is then far larger than the outgoing IP interface MTU size, entailing the sender to perform IP level fragmentation on the TCP packet.

Too low a value will lead to massive inefficiencies and may stall the sender, which could potentially lead to some congestion issues within the sender. However, a well-designed sender can readily cope with

such issues, so a small MSS value is certainly inefficient, but it should not represent any form of attack vector. The recent security notices point to some platform vulnerabilities within the sender that are exposed by low MSS values.

What guidance is there on setting the TCP MSS value?

RFC791 provides IP MTU guidance, stating that:

> "All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams."

Given that IP has no explicit MTU signalling capability, this explicit recommendation of obtaining assurance of the receiver's preparedness to accept larger IP datagrams is presumably referring to the TCP MSS value.

RFC879 provided quite explicit guidance about the TCP MSS value:

> "THE TCP MAXIMUM SEGMENT SIZE IS THE IP MAXIMUM DATAGRAM SIZE MINUS FORTY. The default IP Maximum Datagram Size is 576. The default TCP Maximum Segment Size is 536."

These documents were written prior to the specification of IPv6 of course, and in RFC2460 the following guidance was given for IPv6:

> "When using TCP over IPv6, the MSS must be computed as the maximum packet size minus 60 octets"

It also states that:

> "IPv6 requires that every link in the internet have an MTU of 1280 octets or greater."

Taken together, RFC2460 is asserting that for TCP over IPv6 the MSS value would be expected to be 1220 or greater.

These days the now ancient Ethernet specification still dominates the networking environment (although the old thick yellow coaxial cables and even the CSMA/CD 10Mbps common bus protocol were both been consigned to the networking section of silicon heaven years ago!) This means that most common IP packet MTU is 1500 octets.

Further clarification was provided in RFC6691, TCP Options and Maximum Segment Size (published in July 2012):

> "When calculating the value to put in the TCP MSS option, the MTU value SHOULD be decreased by only the size of the fixed IP and TCP headers and SHOULD NOT be decreased to account for any possible IP or TCP options; conversely, the sender MUST reduce the TCP data length to account for any IP or TCP options that it is including in the packets that it sends. […] the goal is to avoid IP-level fragmentation of TCP packets."

That would imply that the most common anticipated TCP MSS values would correspond to a 1500 octet MTU in both IPv4 and Ipv6. That would imply that we should see a MSS value of 1460 in IPv4 and 1440 in IPv6.

How well does practice line up with the theory?

Measuring TCP MSS values

We looked at the MSS sizes in the HTTP(S) sessions offered by clients who connected to our measurement's servers as part of our large measurement program into IPv6 deployment. We collected all TCP handshakes that occurred between 1 July 2019 and 25 July 2019, and pulled out their MSS value from the SYN packet received form the client.

We saw 2,349,631,475 TCP sessions over this period, and once we removed the duplicate entries for multiple TCP sessions from the same endpoint within a similar timeframe with a common MSS value, we were left with 436,095,954 unique TCP sessions.

Surprisingly enough some 290 endpoints offered an MSS value of 0 and 140 offered a value of 1. A total to 15,274 sessions were opened with MSS values of 500 or lower. This count represents 0.004% of all observed TCP sessions. Small MSS values exist, but they are very much a rarity in the larger population of the Internet. At the other end of the range of observed MSS values, 4 sessions used a value of 65516 (which is the IPv4 maximum MTU minus 40). If we categorise any MSS value greater than 1,460 as some form of *jumbo MSS*, then we observed 55,109 sessions used jumbo MSS values, or 0.013% of all TCP sessions.

> The industry never reached clear agreement on exactly what a *jumbo frame* size was to be, but a value of 9,216 octets has been commonly quoted, as has the Internet2-defined value of 9,000 octets. The lack of agreement within the IEEE to define a single definition of a jumbo frame is not entirely unique, as many media-level protocols have used what could only be described in retrospect as idiosyncratic maximum MTU values. IEEE 802.5 Token Ring used an MTU of up to 4,464 octets, FDDI used 4,532 octets, ATM used 9,180 octets and IEEE 802.11 used 7,935 octets. Perhaps this diversity in media-based MTU values is not all that surprising, and what is perhaps more surprising is an emerging consensus of a commonly assumed MTU of 1,500 octets in the Internet.

The most common observed MSS values are shown in Table 1.

MSS	Count	Proportion
1460	77,911,532	17.9%
1400	70,233,027	16.1%
1370	47,013,384	10.8%
1452	38,914,006	8.9%
1440	37,209,335	8.5%
1360	29,441,242	6.8%
1412	22,738,989	5.2%
1300	18,363,089	4.2%
1380	16,600,330	3.8%
1420	14,785,421	3.4%
1432	8,275,087	1.9%
1410	5,652,204	1.3%
1340	5,035,511	1.2%
1390	3,806,372	0.9%
1358	3,122,473	0.7%
1368	2,778,037	0.6%
1388	2,755,457	0.6%
1350	2,322,741	0.5%
1394	1,729,446	0.4%
1220	1,726,123	0.4%

Table 1 – Most common MSS values

The 1460 value appears to correlate with a 1500 octet MTU and a 40 octet IPv4 and TCP packet header.

We can separate the TCP MSS values used in IPv4 and IPv6, as shown in Table 2.

IPv4 MSS	Count	Proportion		IPv6 MSS	Count	Proportion
1460	77,911,398	22.8%		1370	25,423,389	27.1%
1400	65,318,372	19.1%		1440	19,496,868	20.8%
1452	38,914,004	11.4%		1432	7,772,563	8.3%
1360	28,659,446	8.4%		1300	5,435,839	5.8%
1370	21,589,995	6.3%		1400	4,914,655	5.2%
1412	19,952,380	5.8%		1380	4,612,145	4.9%
1440	17,712,467	5.2%		1340	3,710,855	4.0%
1300	12,927,250	3.8%		1358	2,815,663	3.0%
1420	12,695,442	3.7%		1412	2,786,609	3.0%
1380	11,988,185	3.5%		1368	2,761,014	2.9%
1410	5,514,361	1.6%		1390	2,684,566	2.9%
1388	2,688,249	0.8%		1420	2,089,979	2.2%
1414	1,458,592	0.4%		1220	1,603,225	1.7%
1394	1,413,484	0.4%		1312	1,438,205	1.5%
1340	1,324,656	0.4%		1350	1,372,976	1.5%
1344	1,147,697	0.3%		1362	1,090,572	1.2%
1390	1,121,806	0.3%		1360	781,796	0.8%
1382	1,067,672	0.3%		1426	481,397	0.5%
1240	1,048,430	0.3%		1428	419,016	0.4%
1320	975,281	0.3%		1240	418,657	0.4%

Table 2 – Most common MSS values in IPv4 and IPv6

The 1370 value in IPv6 is somewhat unusual, as it corresponds to an IPv6 MTU of 1430 octets. It appears that it is a common situation to use a 1430 octet MTU in hosts, presumably as such a value (and any value less than 1500 octets) would minimise both the risks of IP fragmentation and path MTU issues that may arise from path element encapsulation. The range of observed MSS values between 1300 and 1440 in both IPv4 and IPv6 points to the existence of a common action of constraining the IP MTU size in order to eliminate IP fragmentation in both IPv4 and IPv6.

> I described the problem back in 2009 in "A Tale of Two Protocols: IPv4, IPv6, MTUs and Fragmentation" (https://www.potaroo.net/ispcol/2009-01/mtu6.html) and pointed out why a reduced MTU setting would be a reasonable response to this problem.

We also saw the MSS value of 536 in 34,473 cases, 32,229 of which were in IPv4 and 2,244 in IPv6.

While there is no particular media type that uses an MTU of 1280, the minimum unfragmented packet MTU size of IPv6 we had observed a minor clustering of MSS values at 1220, with 1,603,225 IPv6 samples using this MSS value.

The overall distribution of observed IPv4 MSS sizes is shown in Figure 1. The plot uses a log scale in both the size and frequency count.

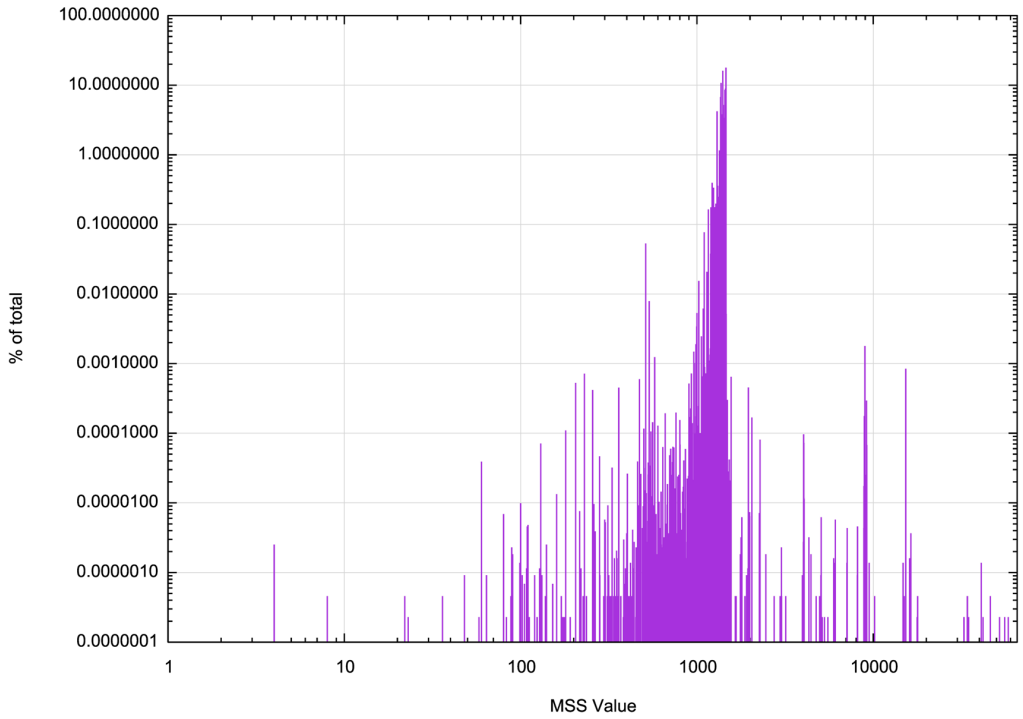

Figure 1 – Relative incidence of observed MSS values

A cumulative frequency distribution can be overlayed on this plot, as shown in Figure 2. The clustering of MSS values between 1000 and 1,500 octets, which encompass some 99.9% of all samples in evident in this distribution.

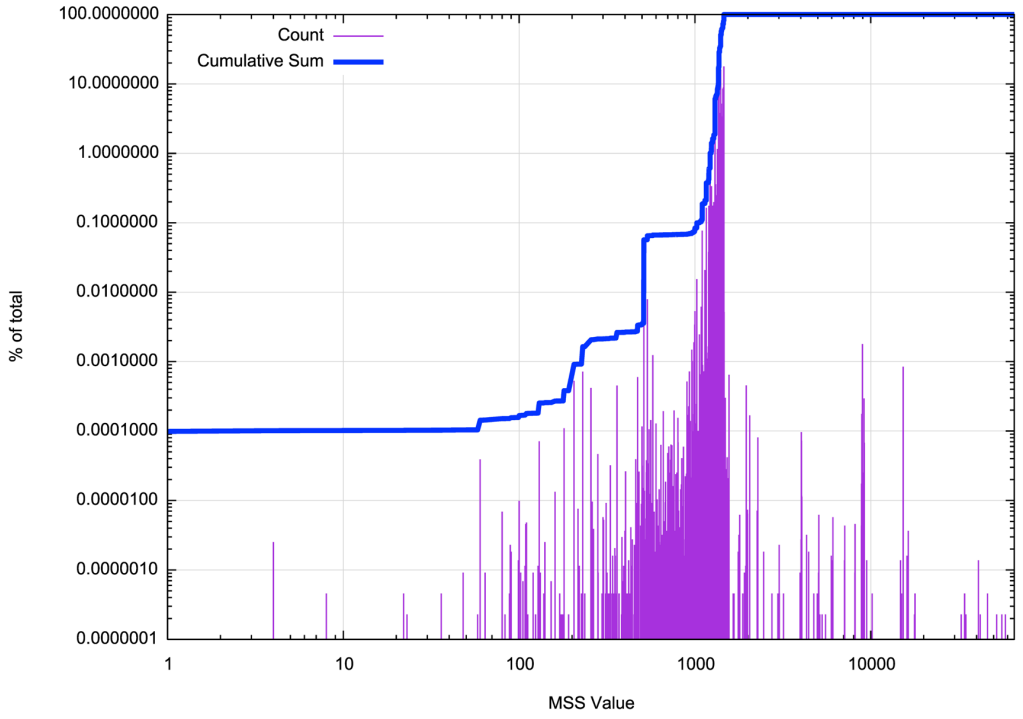

Figure 2 – Cumulative Distribution of MSS values.

The IPv4 and IPv6 values can be plotted separately, as shown in Figure 3. While MSS values greater than 1500 appear to have a similar distribution in both protocols, there are far fewer instances of small MSS values in IPv6 than IPv4.

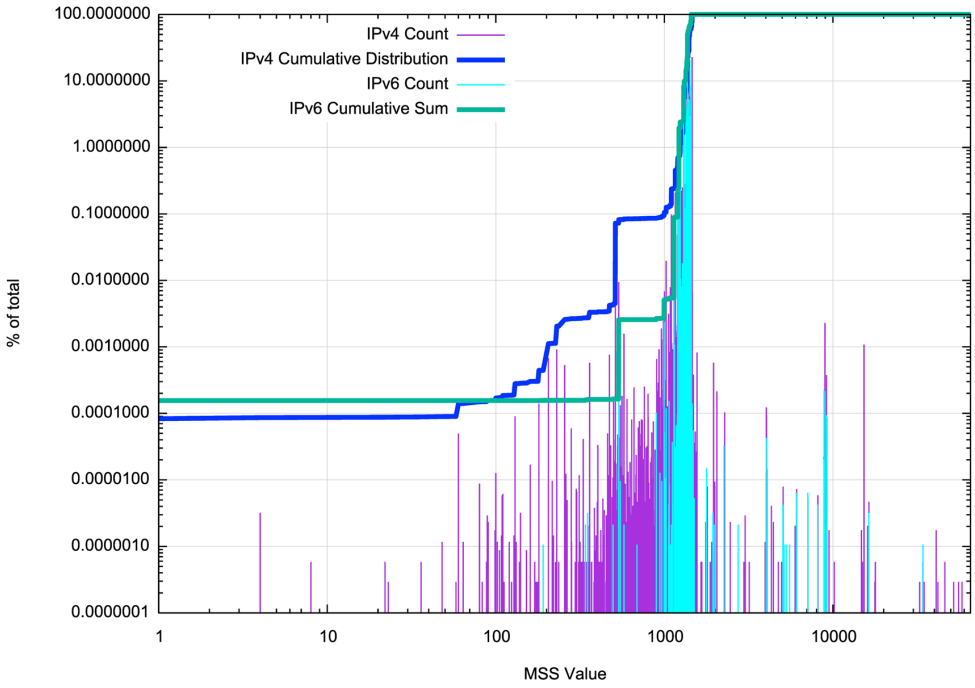

Figure 3 – Cumulative Distribution of MSS values.

Incidence of Low MSS Values

Where might we see hosts with low (less than 500) MSS values?

In IPv4 one half of all such systems are located in Germany and the Netherlands. Adding South Korea, France, Bangladesh and Pakistan to the set encompasses some 80% of all hosts with MSS values less than 500. In the case of IPv6 some 90% of all low MSS values are from hosts located in Germany.

In terms of origin network in IPv4 the networks that contain the most hosts with low MSS values are AS28753, LEASEWEB-DE-FRA-10, and AS60781, LEASEWEB-NL-AMS-01 Netherlands, both operated by Leaseweb, a large hosting enterprise. In IPv6 the majority of instances originate from AS12816, a network which is apparently operated by the Leibniz Supercomputing Centre. It may be that this high incidence of these very low MSS values in these networks could be due to some bug or operational misconfiguration in web hosting equipment, or an unintentional configuration choice made by a client of this virtual system hosting service.

Incidence of High MSS Values

And where are hosts that use large TCP MSS values (values greater than 1460)?

In IPv4 the United States contains 21% of all such hosts, and the somewhat diverse collection of Niger, South Korea, India, Russia and Ireland also each host some 4% of the total count of such hosts. The picture alters with IPv6, with one half of all such hosts located in Japan and a little under one third located in the United States.

At the network level the Amazon ASNs, AS14618 and AS16509 were most commonly found to be hosting high MSS-valued TCP stacks in IPv4, while the ISPs KDDI and NTT's OCN were hosting high-valued MSS hosts in IPv6.

It appears that while some form of hosting or cloud system generates large MSS value in IPv4, some form of configuration of consumer product might be the cause in IPv6.

Incidence of Adjusted MSS values

We can make the supposition that an MSS value of between 1,260 and 1,440 has been the result of a deliberate adjustment of the host MTU value in order to reduce the risk of packet fragmentation and path MTU 'black holes'.

> A *path MTU black hole* occurs when a server emits a packet that is too large for a network path element on the path to the receiver and in the case of IPv4 the packet's Don't Fragment bit has been set or it's an IPv6 packet, and the return path to the server is blocking ICMP messages for some reason.
>
> At this point the connection will stall. The sender is waiting for either an ACK of the data sequence number in the dropped packet or an ICMP packet to indicate that that there is an MTU issue. The ACK will never arrive as the packet has been dropped and the ICMP message has been blocked within the network.
>
> The server will timeout and retransmit the large packet, to no effect. It may do so indefinitely unless some local overall session timeout is in effect, or TCP keepalives are in use.
>
> The client has no outstanding data so it will not retransmit, and it will just hang waiting for a packet that will never arrive. TCP keepalives may identify this hung state and kill the hung TCP session.

We see these adjusted MSS values in those locations with a high IPv6 deployment volume, including India, the United States, Japan and Vietnam.

What values should be used for TCP MSS?

A decade ago, the best advice around was to use a down-adjusted MSS value, such as 1300, 1380 or ever 1400. The reason was to avoid Path MTU issues, particularly when using IPv6, and the reason why path MTU issues were encountered in IPv6 was the prevalent use of IPv6-in-IPv4 encapsulation tunnels in IP transit paths.

I'm not sure that ICMP filtering has got any better or worse in the last decade, but what has improved markedly is the use of 'native' IPv6 transit paths in the Internet.

While it was probably foolhardy to use a 1500 MTU and a 1440 MSS with IPv6 a decade ago it appears now be a reasonable choice. Of course, not every tunnel has been removed and not every potential path MTU issue has been eliminated from the network, but the situation is a whole lot better than it was a decade ago, and the expectation of MTU-related problems is far lower than it was.

If one were to use a 1480 MTU and corresponding TCP MSS values of 1420 in IPv4 and 1400 in IPv6 it would be reasonable to anticipate that the resultant TCP service would be adequately reliable. We could probably take it a little bit further These days I'd be cautiously optimistic that a 1500 MTU would work across the dual stack Internet to the satisfaction of most users.

As for the CVE mitigation advice to refuse a connection attempt when the remote end MSS value is 500 or lower, I'd say that's good advice. It seems that the low MSS values are the result of some form of

misconfiguration or error, and rather than attempting to mask over the error and persisting with an essentially broken TCP connection that is prone to generating a packet deluge, the best option is to just say "no" at the outset. If we all do that, then the misconfiguration will be quickly identified and fixed, rather than being silently masked over.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net