How big was that?

Some observations prompted by CVE-2019-11477

Joel Jaeggli - Fastly

Complex of Vulnerabilties

- Originally discovered by Jonathan Looney / Netflix
  - CVE-2019-11477
  - CVE-2019-11478
  - CVE-2019-11479
- One part of proposed mitigation was:

> Workaround #1: Block connections with a low MSS using one of the supplied filters. (The values in the filters are examples. You can apply a higher or lower limit, as appropriate for your environment.) Note that these filters may break legitimate connections which rely on a low MSS. Also, note that this mitigation is only effective if TCP probing is disabled (that is, the `net.ipv4.tcp_mtu_probing`sysctl is set to 0, which appears to be the default value for that sysctl).
>
> ```
> iptables -A INPUT -p tcp -m tcpmss --mss 1:500 -j DROP
> ip6tables -A INPUT -p tcp -m tcpmss --mss 1:500 -j DROP
> ```

What’s a Legitimate TCP MSS?

- RFC 879

To resolve the ambiguity in the TCP Maximum Segment Size option definition the following rule is established:

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.

- RFC 791

Every internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled

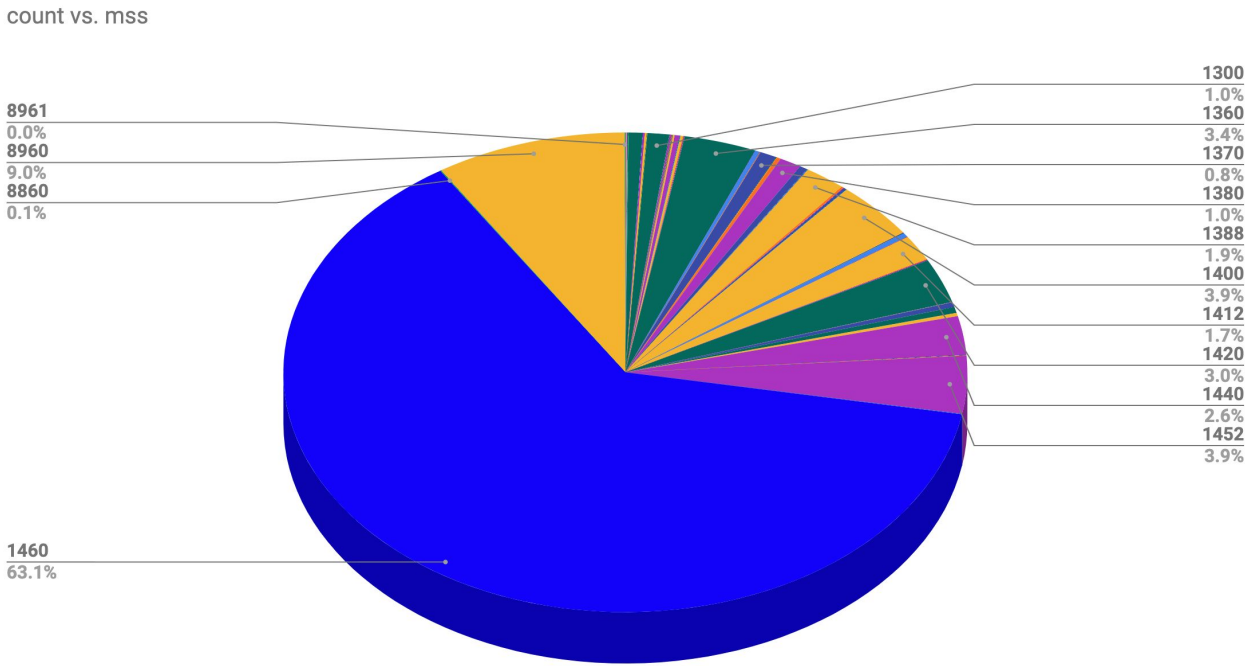
- RFC 2460 ...

What MSS values do we see?

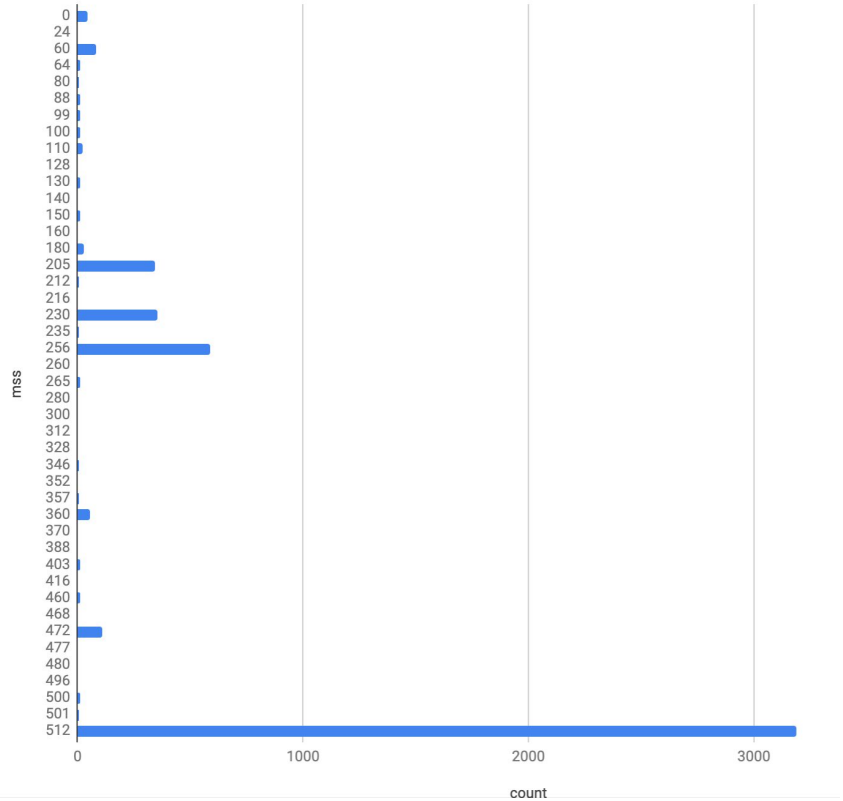
Sample 100M TCP SYNS in 5 markets for our own network.

0 24 60 64 80 88 99 100 110 128 130 140 150 160 180 205 212 216 230 235 256 260 265 280 300 312 328 346 352 357 360 370 388 403 416 460 468 472 477 480 496 500 501 512 528 532 **536** 537 538 540 542 545 550 552 556 560 576 578 586 600 607 610 617 626 628 639 640 660 680 700 706 710 711 712 716 717 718 720 725 726 734 740 745 748 750 752 760 764 772 780 785 788 790 800 801 808 810 820 828 836 843 844 848 852 854 859 860 863 866 870 880 881 882 889 890 900 906 908 910 912 916 918 919 920 922 930 935 938 939 940 942 952 956 959 960 961 962 964 965 966 970 975 976 979 980 982 984 986 988 989 990 992 995 996 1000 1001 1005 1006 1008 1010 1011 1012 1017 1018 1020 1022 1024 1026 1032 1035 1036 1038 1039 1040 1041 1043 1048 1050 1051 1052 1054 1058 1060 1061 1062 1063 1064 1068 1070 1071 1072 1078 1079 1080 1081 1082 1083 1084 1085 1086 1088 1090 1092 1094 1095 1096 1099 1100 1101 1102 1103 1104 1105 1106 1108 1109 1110 1111 1112 1119 1120 1122 1124 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1144 1146 1147 1148 1150 1151 1152 1153 1154 1155 1156 1158 1159 1160 1161 1163 1164 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1179 1180 1181 1182 1183 1184 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 **1240** 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 **1440** 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 **1460** 1464 1468 1470 1474 1478 1480 1482 1484 1488 1496 1500 1502 1506 1510 1512 1520 1540 1546 1552 1556 1558 1560 1660 1668 1798 1862 1894 1956 1960 1990 1994 2004 2008 2048 2252 2278 2760 2960 4024 4030 4034 4038 4042 4056 4430 4460 5062 5066 5960 6033 6090 7110 7114 7378 7960 8052 8102 8134 8152 8460 8860 8861 8872 8873 8880 8910 8911 8932 8933 8940 8956 8960 8961 8974 8982 9060 9078 9122 9126 9130 9138 9146 9152 9153 9158 9162 9172 9176 9560 10178 10180 14760 15284 31749 47960 63910 63960 65420 65496

Percentages, MSS sizes as sampled on 100M syns



Frequency of small Oddballs for 100M Samples



What problems were exposed?

- CVE-2019-114xx points to a generalized problem of poorly understood or infrequequently exercised code paths.
  - Other Examples exist:
    - CVE-2013-7470 - ip options handling
    - CVE-2016-10142 - IPv6 PTB handling
- Implementers or users employing configurations that are notionaly feasible, but not necessarily compliant with a specification.
  - MSS < 536 or < 1240
  - Obviously header plus 0 payload bounds the lowest feasible packet size.
- Malicious clients are able to control the behavior of servers.
- There is a need to mitigate certain kinds of behavior either temporarily or indefinitely, without gratuitously breaking users with bad assumptions.

Problems - RFC 6691

- Is there advice that we should be offering here?
  - Consider what we did in RFC 6691 3.1

> ...
>
> When TCP is used in a situation when either the IP or TCP headers are not minimum and yet the maximum IP datagram that can be received remains 576 octets then the TCP Maximum Segment Size option must be used to reduce the limit on data octets allowed in a TCP segment.
>
> For example, if the IP Security option (11 octets) were in use and the IP maximum datagram size remained at 576 octets, then the TCP should send the MSS with a value of 525 (536-11).
>
> That is incorrect. The simpler and more correct statement would be:
>
> When TCP is used in a situation where either the IP or TCP headers are not minimum, the sender must reduce the amount of TCP data in any given packet by the number of octets used by the IP and TCP options.
>
> ...

Problems - Sane Values

- The linux kernel now imposes a minimum.
  - https://cdn.kernel.org/pub/linux/kernel/v4.x/ChangeLog-4.19.52
  - `48 (TCP_MIN_SND_MSS)`
  - How high should we set this?
- A reading of our documents suggests that you would be well within the specification to enforce the sending of TCP packets accounting for a minimum MTU of 576 (IPv4) and 1280 (IPv6). If someone has a link MTU smaller, too bad.
- If these devices actually have link MTUs that will not accommodate, a 576 or 1280 byte IP packet they will break.

Overflow Slides

Problems - Sanity, high values.

- RFC 6691 supposes interfaces with variable MTUs. In fact, mostly this is done by routes. Very large MSS values seem like an invitation to negotiate to the highest feasible MTU. Is that why those appear?

Regional, or network specific concentration of problem clients

- Initially I sampled in one pop.
- Concentrations of clients exhibiting specific behavior, were only revealed when more pops were included in the sample.
  - For me this implies that at least some of the pockets of weirdness are concentrated and not generalizable.
- Are these pockets of weirdness, non-internet leakage. IOT applications, other stuff?

Relationship of link mtu to advertised MSS

- It’s my assumption that in not all of these cases is the link MTU actually related to the advertised MSS.
- If some of them are will these MTUs also appear in the case of QUIC?
  - QUIC effectively requires a minimum of 1200 bytes
  - https://tools.ietf.org/html/draft-ietf-quic-transport-22#section-14.1