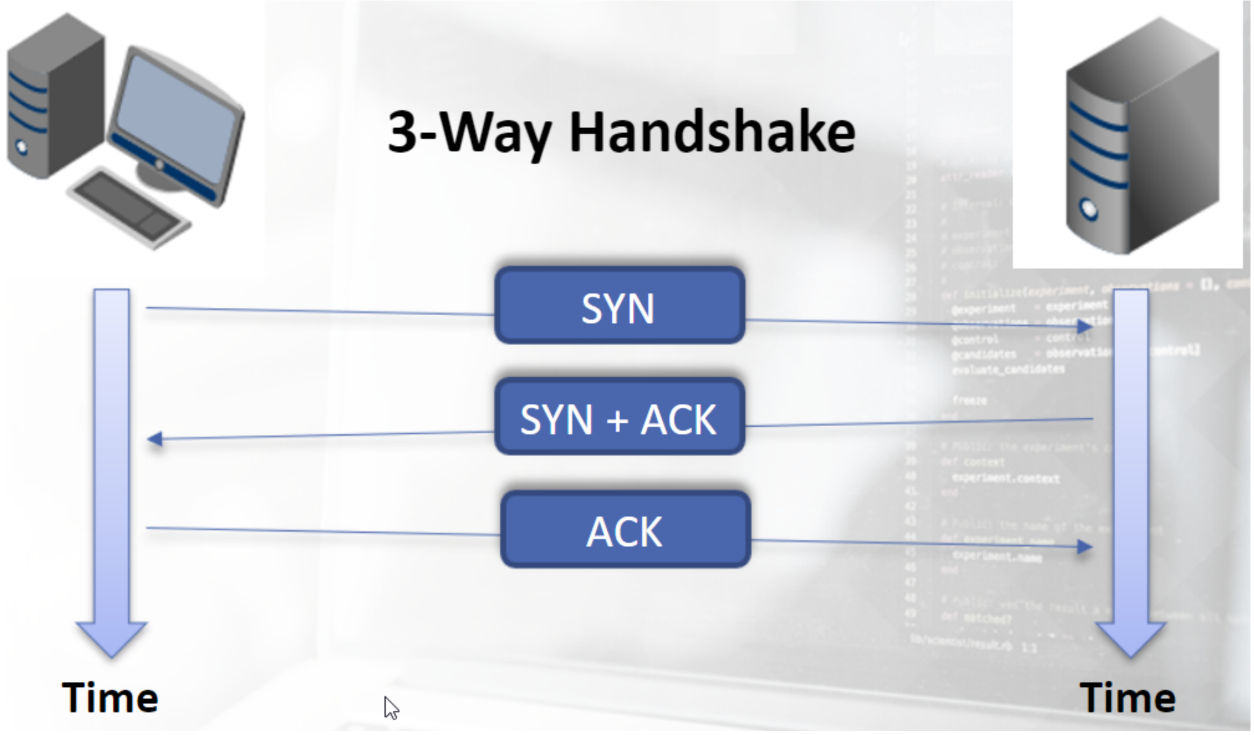
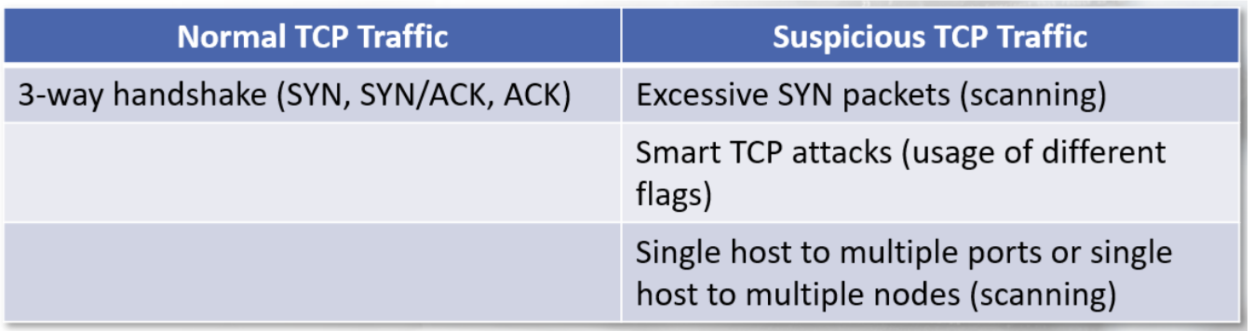
TCP traffic:

TCP stands for Transmission Control Protocol, it ensures delivery of data from source node to the detonation node. TCP handles sequencing of packets and error recovery.

TCP before start sending the data, it will initiate a 3 Way handshake between the 2 nodes:



Distinguish Normal and Suspicious TCP traffic:



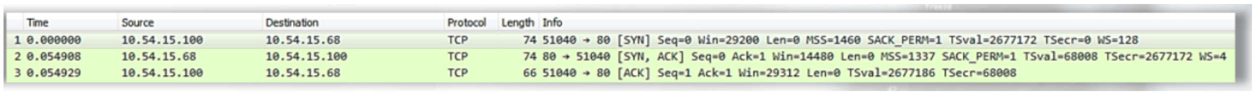
Normal TCP Traffic	Suspicious TCP Traffic
3-way handshake (SYN, SYN/ACK, ACK)	Excessive SYN packets (scanning)
	Smart TCP attacks (usage of different flags)
	Single host to multiple ports or single host to multiple nodes (scanning)

If we see too many SYN packets are send then it indicate scanning, as we know the 3 way handshake starts with SYN, and the attacker will send SYN packet to all IPs within the network to see which IPs will respond and those that respond, the attacker will know these IPs are active.

Attackers can manipulate the packet like change the flags within it to do attacks.

Or a host communicate to so many ports in host, by sending SYN packet to each of its Port to see if the port is open or not. (Port scanning)

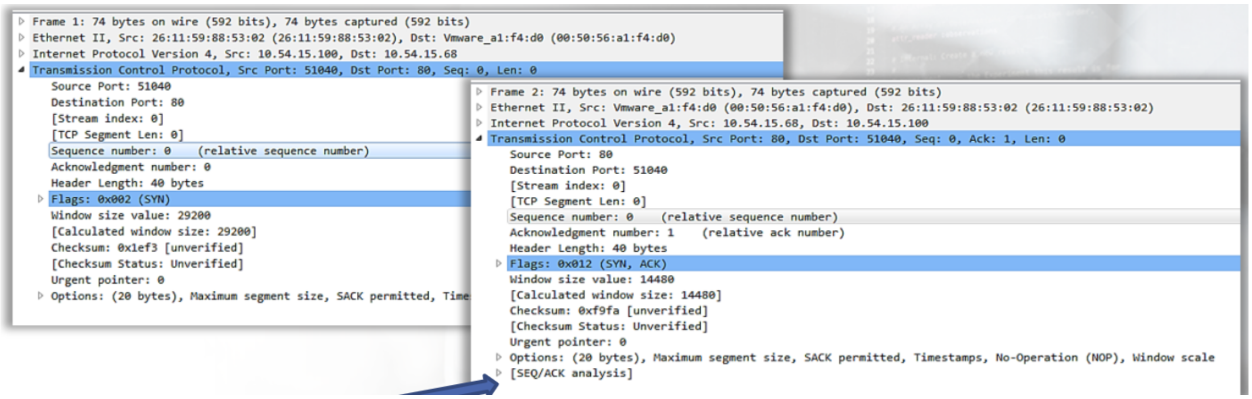
An example of 3 way handshake:



```
  Time        Source          Destination       Protocol  Length  Info
1 0.000000    10.54.15.100    10.54.15.68       TCP         74 51040 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2677172 TSecr=0 WS=128
2 0.054908    10.54.15.68     10.54.15.100      TCP         74 80 → 51040 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=68008 TSecr=2677172 WS=4
3 0.054929    10.54.15.100    10.54.15.68       TCP         66 51040 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2677186 TSecr=68008
```

Normal TCP Traffic:

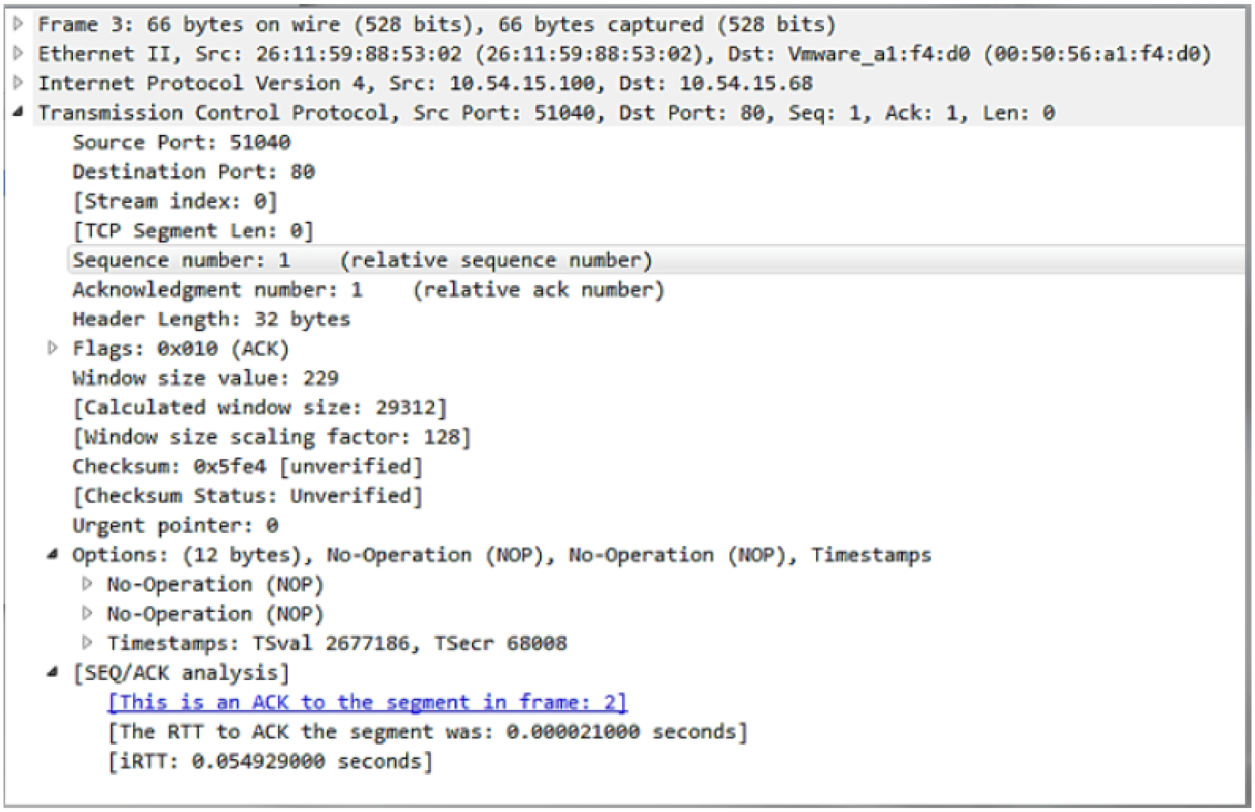
Here is the packets of the handshake from the screen shot above:



```
▷ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▷ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0)
▷ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
◢ Transmission Control Protocol, Src Port: 51040, Dst Port: 80, Seq: 0, Len: 0
     Source Port: 51040
     Destination Port: 80
     [Stream index: 0]
     [TCP Segment Len: 0]
     Sequence number: 0    (relative sequence number)
     Acknowledgment number: 0
     Header Length: 40 bytes
   ▷ Flags: 0x002 (SYN)
     Window size value: 29200
     [Calculated window size: 29200]
     Checksum: 0x1ef3 [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▷ Options: (20 bytes), Maximum segment size, SACK permitted, Time
```

```
▷ Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▷ Ethernet II, Src: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0), Dst: 26:11:59:88:53:02 (26:11:59:88:53:02)
▷ Internet Protocol Version 4, Src: 10.54.15.68, Dst: 10.54.15.100
◢ Transmission Control Protocol, Src Port: 80, Dst Port: 51040, Seq: 0, Ack: 1, Len: 0
     Source Port: 80
     Destination Port: 51040
     [Stream index: 0]
     [TCP Segment Len: 0]
     Sequence number: 0    (relative sequence number)
     Acknowledgment number: 1    (relative ack number)
     Header Length: 40 bytes
   ▷ Flags: 0x012 (SYN, ACK)
     Window size value: 14480
     [Calculated window size: 14480]
     Checksum: 0xf9fa [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▷ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
   ▷ [SEQ/ACK analysis]
```

We see the Frame 1 which indicates the first packet, so we see it has source, destination port, acknowledgment number… and see the flag SYN which sent the first thing in the 3 way handshake by the host that wants to connect to another host. And in the 2nd packet we see SYN/ACK flags, which is the respond to the SYN packet.



```
▷ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▷ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0)
▷ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
◢ Transmission Control Protocol, Src Port: 51040, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 51040
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header Length: 32 bytes
  ▷ Flags: 0x010 (ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x5fe4 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ◢ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▷ No-Operation (NOP)
    ▷ No-Operation (NOP)
    ▷ Timestamps: TSval 2677186, TSecr 68008
  ◢ [SEQ/ACK analysis]
       [This is an ACK to the segment in frame: 2]
       [The RTT to ACK the segment was: 0.000021000 seconds]
       [iRTT: 0.054929000 seconds]
```

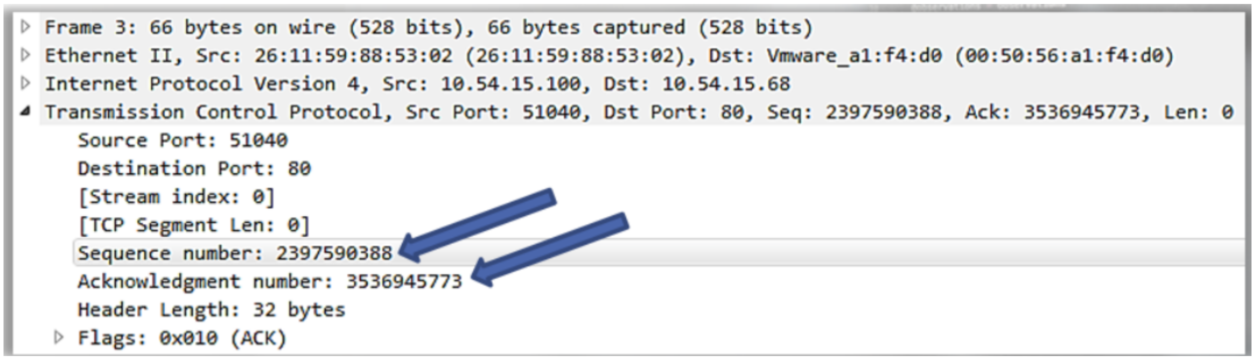
This is the 3 packet of the 3 way handshake. We see it has the ACK flag set int it.

We should pay attention to the Sequence number and Acknowledgement number when analyzing TCP packets.

Wireshark by default will keep track of all TCP sessions and convert the Sequence numbers and Acknowledgment number into relative numbers, mean instead o displaying the real/absolute SEQ and ACK number numbers, it will display SEQ and ACK number relative to the first seen segment for the conversation, and this mean that all SEQ and Ack numbers will start at 0 for the first packet of the each conversation or session. This makes the numbers easier to read then being randomly select from 2^32.

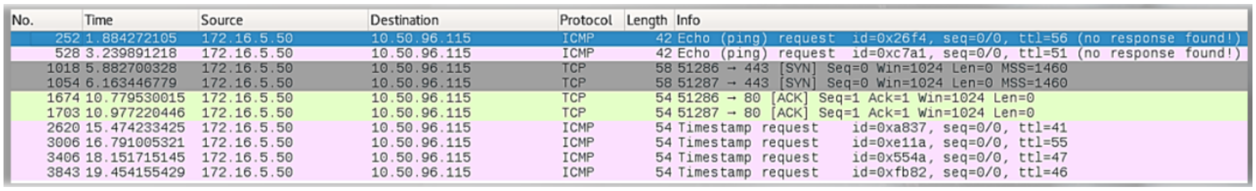
To enable/disable this we go to this settings : Edit > Preferences > Protocols > TCP > Relative Sequence numbers . to enable check the box for disable uncheck the box:

If disabled this is how it looks like:



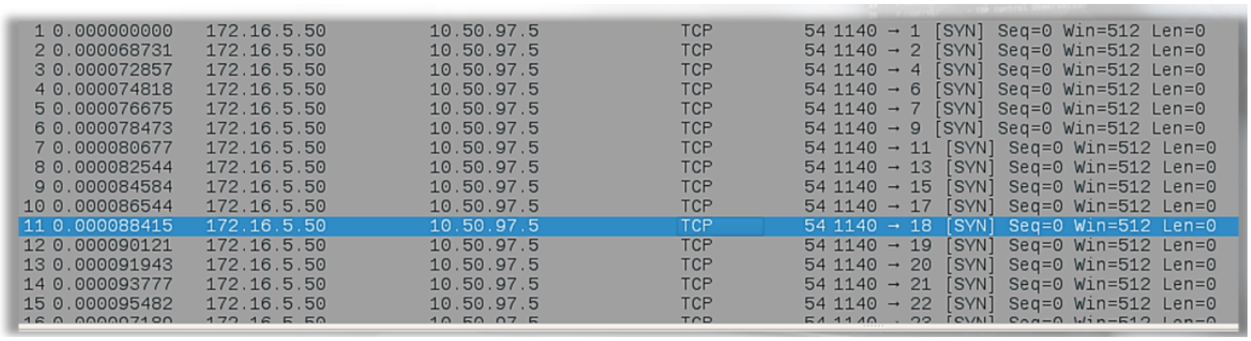
```
▷ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▷ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: Vmware_a1:f4:d0 (00:50:56:a1:f4:d0)
▷ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.68
◢ Transmission Control Protocol, Src Port: 51040, Dst Port: 80, Seq: 2397590388, Ack: 3536945773, Len: 0
     Source Port: 51040
     Destination Port: 80
     [Stream index: 0]
     [TCP Segment Len: 0]
     Sequence number: 2397590388
     Acknowledgment number: 3536945773
     Header Length: 32 bytes
   ▷ Flags: 0x010 (ACK)
```

Suspicious TCP traffic:



```
No.   Time         Source       Destination    Protocol Length Info
  252 1.884272105  172.16.5.50  10.50.96.115   ICMP     42 Echo (ping) request  id=0x26f4, seq=0/0, ttl=56 (no response found!)
  528 3.239891218  172.16.5.50  10.50.96.115   ICMP     42 Echo (ping) request  id=0xc7a1, seq=0/0, ttl=51 (no response found!)
 1018 5.882700328  172.16.5.50  10.50.96.115   TCP      58 51286 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 1054 6.163446779  172.16.5.50  10.50.96.115   TCP      58 51287 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
 1674 10.779530015 172.16.5.50  10.50.96.115   TCP      54 51286 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
 1703 10.977220446 172.16.5.50  10.50.96.115   TCP      54 51287 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
 2620 15.474233425 172.16.5.50  10.50.96.115   ICMP     54 Timestamp request    id=0xa837, seq=0/0, ttl=41
 3006 16.791005321 172.16.5.50  10.50.96.115   ICMP     54 Timestamp request    id=0xe11a, seq=0/0, ttl=55
 3406 18.151715145 172.16.5.50  10.50.96.115   ICMP     54 Timestamp request    id=0x554a, seq=0/0, ttl=47
 3843 19.454155429 172.16.5.50  10.50.96.115   ICMP     54 Timestamp request    id=0xfb82, seq=0/0, ttl=46
```

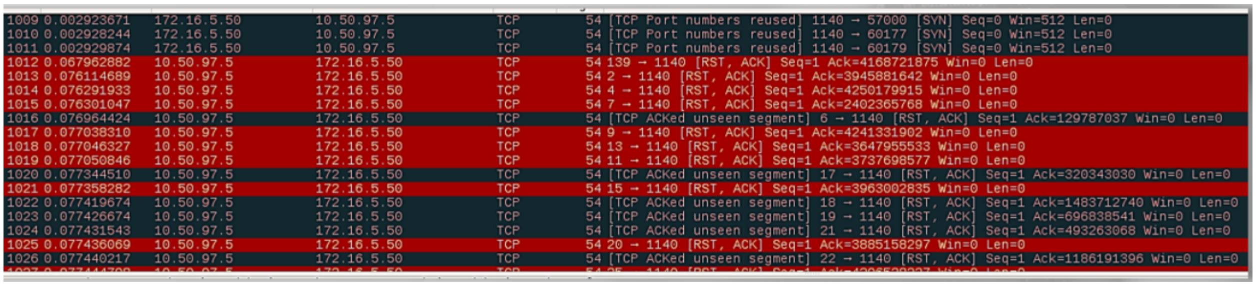
Here a ping sweep is performed using Nmap, this ping sweep used Echo (Ping) request and TCP SYN connection to port 80 and 443. And one thing we mentioned before that to look for ICMP Timestamp requests after a Echo ping request, that mean if someone send a ICMP echo request and then after it a ICMP Timestamp request is sent, that will look suspicious.



```
 1 0.000000000   172.16.5.50   10.50.97.5   TCP   54 1140 → 1 [SYN] Seq=0 Win=512 Len=0
 2 0.000068731   172.16.5.50   10.50.97.5   TCP   54 1140 → 2 [SYN] Seq=0 Win=512 Len=0
 3 0.000072857   172.16.5.50   10.50.97.5   TCP   54 1140 → 4 [SYN] Seq=0 Win=512 Len=0
 4 0.000074818   172.16.5.50   10.50.97.5   TCP   54 1140 → 6 [SYN] Seq=0 Win=512 Len=0
 5 0.000076675   172.16.5.50   10.50.97.5   TCP   54 1140 → 7 [SYN] Seq=0 Win=512 Len=0
 6 0.000078473   172.16.5.50   10.50.97.5   TCP   54 1140 → 9 [SYN] Seq=0 Win=512 Len=0
 7 0.000080677   172.16.5.50   10.50.97.5   TCP   54 1140 → 11 [SYN] Seq=0 Win=512 Len=0
 8 0.000082544   172.16.5.50   10.50.97.5   TCP   54 1140 → 13 [SYN] Seq=0 Win=512 Len=0
 9 0.000084584   172.16.5.50   10.50.97.5   TCP   54 1140 → 15 [SYN] Seq=0 Win=512 Len=0
10 0.000086544   172.16.5.50   10.50.97.5   TCP   54 1140 → 17 [SYN] Seq=0 Win=512 Len=0
11 0.000088415   172.16.5.50   10.50.97.5   TCP   54 1140 → 18 [SYN] Seq=0 Win=512 Len=0
12 0.000090121   172.16.5.50   10.50.97.5   TCP   54 1140 → 19 [SYN] Seq=0 Win=512 Len=0
13 0.000091943   172.16.5.50   10.50.97.5   TCP   54 1140 → 20 [SYN] Seq=0 Win=512 Len=0
14 0.000093777   172.16.5.50   10.50.97.5   TCP   54 1140 → 21 [SYN] Seq=0 Win=512 Len=0
15 0.000095482   172.16.5.50   10.50.97.5   TCP   54 1140 → 22 [SYN] Seq=0 Win=512 Len=0
16 0.000097180   172.16.5.50   10.50.97.5   TCP   54 1140 → 23 [SYN] Seq=0 Win=512 Len=0
```

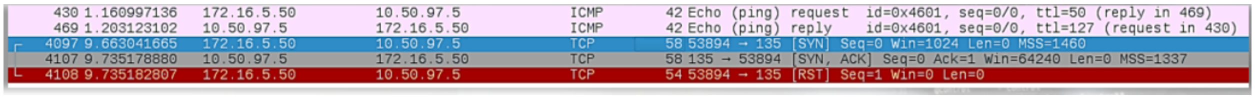
Here we see many TCP SYN packets are send without getting response of SYN,ACK and it should a rise a RED flag if we see soo many SYN packets sent within the network, here we all those SYN packets send to one host but the port number increments, so its Port scanning.

Wireshark has feature of Highlighting, that can help us in analyzing traffic easier:



```
1009 0.002923671   172.16.5.50     10.50.97.5      TCP      54 [TCP Port numbers reused] 1140 → 57000 [SYN] Seq=0 Win=512 Len=0
1010 0.002928244   172.16.5.50     10.50.97.5      TCP      54 [TCP Port numbers reused] 1140 → 60177 [SYN] Seq=0 Win=512 Len=0
1011 0.002929874   172.16.5.50     10.50.97.5      TCP      54 [TCP Port numbers reused] 1140 → 60179 [SYN] Seq=0 Win=512 Len=0
1012 0.067962882   10.50.97.5      172.16.5.50     TCP      54 139 → 1140 [RST, ACK] Seq=1 Ack=4168721875 Win=0 Len=0
1013 0.076114689   10.50.97.5      172.16.5.50     TCP      54 2 → 1140 [RST, ACK] Seq=1 Ack=3945881642 Win=0 Len=0
1014 0.076291933   10.50.97.5      172.16.5.50     TCP      54 4 → 1140 [RST, ACK] Seq=1 Ack=4250179915 Win=0 Len=0
1015 0.076301047   10.50.97.5      172.16.5.50     TCP      54 7 → 1140 [RST, ACK] Seq=1 Ack=2402365768 Win=0 Len=0
1016 0.076964424   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 6 → 1140 [RST, ACK] Seq=1 Ack=129787037 Win=0 Len=0
1017 0.077038310   10.50.97.5      172.16.5.50     TCP      54 9 → 1140 [RST, ACK] Seq=1 Ack=4241331902 Win=0 Len=0
1018 0.077046327   10.50.97.5      172.16.5.50     TCP      54 13 → 1140 [RST, ACK] Seq=1 Ack=3647955533 Win=0 Len=0
1019 0.077050846   10.50.97.5      172.16.5.50     TCP      54 11 → 1140 [RST, ACK] Seq=1 Ack=3737698577 Win=0 Len=0
1020 0.077344510   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 17 → 1140 [RST, ACK] Seq=1 Ack=320343030 Win=0 Len=0
1021 0.077358282   10.50.97.5      172.16.5.50     TCP      54 15 → 1140 [RST, ACK] Seq=1 Ack=3963002835 Win=0 Len=0
1022 0.077419674   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 18 → 1140 [RST, ACK] Seq=1 Ack=1483712740 Win=0 Len=0
1023 0.077426674   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 19 → 1140 [RST, ACK] Seq=1 Ack=696838541 Win=0 Len=0
1024 0.077431543   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 21 → 1140 [RST, ACK] Seq=1 Ack=493263068 Win=0 Len=0
1025 0.077436069   10.50.97.5      172.16.5.50     TCP      54 20 → 1140 [RST, ACK] Seq=1 Ack=3885158297 Win=0 Len=0
1026 0.077440217   10.50.97.5      172.16.5.50     TCP      54 [TCP ACKed unseen segment] 22 → 1140 [RST, ACK] Seq=1 Ack=1186191396 Win=0 Len=0
```

here is another port scan:



```
  430 1.160997136   172.16.5.50     10.50.97.5      ICMP      42 Echo (ping) request  id=0x4601, seq=0/0, ttl=50 (reply in 469)
  469 1.203123102   10.50.97.5      172.16.5.50     ICMP      42 Echo (ping) reply    id=0x4601, seq=0/0, ttl=127 (request in 430)
┌ 4097 9.663041665  172.16.5.50     10.50.97.5      TCP       58 53894 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
│ 4107 9.735178880  10.50.97.5      172.16.5.50     TCP       58 135 → 53894 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1337
└ 4108 9.735182807  172.16.5.50     10.50.97.5      TCP       54 53894 → 135 [RST] Seq=1 Win=0 Len=0
```

Focus on the last 3 packets, we see:

- SYN to start the communication on port 135
- SYN,ACK response which indicates that the port 135 is open as we got the SYN,ACK if it wasn't open would have got RST
- RST is sent to end the communication, so the attacker now know the port is open so it doesn't want to send anything more.