Beginners Guide to TShark (Part 2)

February 19, 2020 By Raj Chandel

In the previous article, we learned about the basic functionalities of this wonderful tool called TShark. If you haven't read it until now. Click here.

TL; DR

In this part, we will the Statistical Functionalities of TShark. We will understand different ways in which we can sort our traffic capture so that we can analyse it faster and effectively.

Table of Content

Statistical Options

TShark collects different types of Statistics and displays their result after finishing the reading of the captured file. To accomplish this, we will be using the "-z" parameter with TShark. Initially, to learn about all the different options inside the "-z" parameter, we will be running the TShark with the "-z" parameter followed by the help keyword. This gives us an exhaustive list of various supported formats as shown in the image given below.

```
root@kali:~# tshark -z help
Running as user "root" and group "root". This could be dangerous.
tshark: The available statistics for the "-z" option are:
     afp,srt
     ancp,tree
     ansi_a,bsmap
     ansi_a,dtap
     ansi_map
     bacapp_instanceid,tree
     bacapp_ip,tree
     bacapp_objectid,tree
     bacapp_service,tree
     camel,counter
     camel,srt
     collectd,tree
     conv,bluetooth
     conv,eth
     conv,fc
     conv,fddi
     conv,ip
     conv,ipv6
     conv,ipx
     conv,jxta
     conv,mptcp
     conv,ncp
     conv,rsvp
     conv,sctp
     conv,sll
     conv,tcp
     conv,tr
     conv,udp
     conv,usb
     conv,wlan
     dcerpc,srt
     dests,tree
     dhcp,stat
     diameter,avp
     diameter,srt
     dns,tree
     endpoints,bluetooth
```

Protocol Hierarchy Statistics

Using the TShark we can create a Protocol based Hierarchy Statistics listing the number of packets and bytes using the “io,phs” option in the “-z” parameter. In the case where no filter is given after the “io,phs” option, the statistics will be calculated for all the packets in the scope. But if a specific filter is provided than the TShark will calculate statistics for those packets that match the filter provided by the user. For our demonstration, we first captured some traffic and wrote the contents on a pcap file using the techniques that we learned in part 1 of this article series. Then we will be taking the traffic from the file, and then sort the data into a Protocol Hierarchy. Here we can observe that we have the frames count, size of packets in bytes and the Protocol used for the transmission.

```
tshark  -r wlan.pcap -z io,phs
```

```
===================================================================
Protocol Hierarchy Statistics
Filter:

radiotap                                 frames:66690 bytes:15014549
  wlan_radio                             frames:66690 bytes:15014549
    wlan                                 frames:66690 bytes:15014549
      wlan                               frames:6873 bytes:1923747
      data                               frames:14539 bytes:9494059
      llc                                frames:3158 bytes:1295577
        eapol                            frames:6 bytes:1162
        ipv6                             frames:16 bytes:2136
          icmpv6                         frames:16 bytes:2136
        ip                               frames:3124 bytes:1291079
          udp                            frames:143 bytes:25311
            dhcp                         frames:6 bytes:2448
            dns                          frames:126 bytes:21131
            ntp                          frames:3 bytes:444
            mdns                         frames:8 bytes:1288
          icmp                           frames:2 bytes:240
          tcp                            frames:2979 bytes:1265528
            tls                          frames:781 bytes:455386
              tcp.segments               frames:74 bytes:60600
                tls                      frames:62 bytes:53122
            http                         frames:248 bytes:123041
              data-text-lines            frames:9 bytes:6487
                tcp.segments             frames:4 bytes:2696
              image-jfif                 frames:6 bytes:4156
                tcp.segments             frames:6 bytes:4156
              image-gif                  frames:2 bytes:1352
              tcp.segments               frames:3 bytes:1402
            data                         frames:2 bytes:2924
            tcp.segments                 frames:1 bytes:1462
            _ws.malformed                frames:5 bytes:2666
        arp                              frames:12 bytes:1200
===================================================================
```

Read Filter Analysis

During the first pass analysis of the packet, the specified filter (which uses the syntax of read/display filters, rather than that of capture filters) has to be applied. Packets which are not matching the filter are not considered for future passes. This parameter makes sense with multiple passes. Note that forward-looking fields such as ‘response in frame #’ cannot be used with this filter since they will not have been calculated when this filter is applied. The “-2” parameter performs a two-pass analysis. This causes TShark to buffer output until the entire first pass is done, but allows it to fill in fields that require future knowledge, it also permits reassembly frame dependencies to be calculated correctly. Here we can see two different analysis one of them is first-pass analysis and the latter is the two-pass analysis.

```
tshark -r wlan.pcap -z io,phs,udp -q
tshark -r wlan.pcap -z io,phs -q -2 -R udp
```

```
root@kali:~# tshark -r wlan.pcap -z io,phs,udp -q
Running as user "root" and group "root". This could be dangerous.

===================================================================
Protocol Hierarchy Statistics
Filter: udp

radiotap                                 frames:143 bytes:25311
  wlan_radio                             frames:143 bytes:25311
    wlan                                 frames:143 bytes:25311
      llc                                frames:143 bytes:25311
        ip                               frames:143 bytes:25311
          udp                            frames:143 bytes:25311
            dhcp                         frames:6 bytes:2448
            dns                          frames:126 bytes:21131
            ntp                          frames:3 bytes:444
            mdns                         frames:8 bytes:1288
===================================================================
root@kali:~# tshark -r wlan.pcap -z io,phs -q -2 -R udp
Running as user "root" and group "root". This could be dangerous.

===================================================================
Protocol Hierarchy Statistics
Filter:

radiotap                                 frames:143 bytes:25311
  wlan_radio                             frames:143 bytes:25311
    wlan                                 frames:143 bytes:25311
      llc                                frames:143 bytes:25311
        ip                               frames:143 bytes:25311
          udp                            frames:143 bytes:25311
            dhcp                         frames:6 bytes:2448
            dns                          frames:126 bytes:21131
            ntp                          frames:3 bytes:444
            mdns                         frames:8 bytes:1288
===================================================================
```

Endpoints Analysis

Our next option which helps us with the statistics is the “endpoints”. It will create a table that will list all endpoints that could be seen in the capture. The type function which can be used with the endpoint option will specify the endpoint type for which we want to generate the statistics.

The list of Endpoints that are supported by TShark is:

Sno.	Filter	Description
1	**“bluetooth”**	Bluetooth Addresses
2	**“eth”**	Ethernet Addresses
3	**“fc”**	Fiber Channel Addresses
4	**“fddi”**	FDDI Addresses
5	**“ip”**	IPv4 Addresses
6	**“ipv6”**	IPv6 Addresses
7	**“ipx”**	IPX Addresses
8	**“jxta”**	JXTS Addresses
9	**“ncp”**	NCP Addresses
10	**“rsvp”**	RSVP Addresses
11	**“sctp”**	SCTP Addresses

12	**“tcp”**	TCP/IP socket pairs Both IPv4 and IPv6 supported
13	**“tr”**	Token Ring Addresses
14	**“usb”**	USB Addresses
15	**“udp”**	UDP/IP socket pairs Both IPv4 and IPv6 supported
16	**“wlan”**	IEEE 802.11 addresses

In case that we have specified the filter option then the statistics calculations are done for that particular specified filter. The table like the one generated in the image shown below is generated by picking up single line form each conversation and displayed against the number of packets per byte in each direction as well as the total number of packets per byte. This table is by default sorted according to the total number of frames.

```
tshark -r wlan.pcap -z endpoints,wlan -q | head
```

```
root@kali:~# tshark -r wlan.pcap -z endpoints,wlan -q | head
Running as user "root" and group "root". This could be dangerous.
================================================================================
IEEE 802.11 Endpoints
Filter:<No Filter>
                  |  Packets  |  |  Bytes  |  | Tx Packets |  | Tx Bytes |  | Rx Packets
AsustekC_c3:5e:01      18320       9311075        9843         8435055         8477
Tp-LinkT_16:87:18       8962       1644801        4024         1124143         4938
D-LinkIn_5f:81:6b       8122        950847          50            5484         8072
Motorola_31:a0:3b       8079       2137351        6262         1139916         1817
Tp-LinkT_09:7f:d3       7894       6218261        2930          453787         4964
Broadcast               6444       1728228          18            1164         6426
```

Conversation Analysis

Let’s move on to the next option which is quite similar to the previous option. It helps us with the statistics is the “conversation”. It will create a table that will list all conversation that could be seen in the capture. The type function which can be used with the conversation option will specify the conversation type for which we want to generate the statistics.

If we have specified the filter option then the statistics calculations are done for that particular specified filter. The table generated by picking up single line form each conversation and displayed against the number of packets per byte in each direction, the total number of packets per byte as well as the direction of the conversation travel. This table is by default sorted according to the total number of frames.

```
tshark -r wlan.pcap -z conv,wlan -q | head
```

```
root@kali:~# tshark -r wlan.pcap -z conv,wlan -q | head
Running as user "root" and group "root". This could be dangerous.
================================================================================
IEEE 802.11 Conversations
Filter:<No Filter>
                                               |       ←       |       →       |     Total     |
                                               | Frames  Bytes | Frames  Bytes | Frames  Bytes |
AsustekC_c3:5e:01    ↔ Tp-LinkT_09:7f:d3       |   2841 441682 |   4753 5753274 |   7594 6194956 | 15
Motorola_31:a0:3b    ↔ D-LinkIn_5f:81:6b       |      3    431 |   3455  696782 |   3458  697213 |
Motorola_31:a0:3b    ↔ Tp-LinkT_16:87:18       |   1566 937369 |   1689  358208 |   3255 1295577 | 15
AsustekC_c3:5e:01    ↔ IntelCor_96:a1:a9       |    721  91683 |   2372 2046278 |   3093 2137961 | 15
00:51:88:31:a0:3b    ↔ D-LinkIn_5f:81:6b       |      0      0 |   2898  144900 |   2898  144900 |
```

Expert Mode Analysis

The TShark Statistics Module have an Expert Mode. It collects a huge amount of data based on Expert Info and then prints this information in a specific order. All this data is grouped in the sets of severity like Errors, Warnings, etc., We can use the expert mode with a particular protocol as well. In that case, it will display all the expert items of that particular protocol.

```
tshark -r wlan.pcap -z expert -q | head
```

```
root@kali:~# tshark -r wlan.pcap -z expert -q | head
Running as user "root" and group "root". This could be dangerous.

Errors (5)
==========
   Frequency      Group          Protocol  Summary
           5  Malformed               TCP  New fragment overlaps old data (retransmission?)

Warns (53821)
=============
   Frequency      Group          Protocol  Summary
       13373 Assumption       802.11 Radio  No plcp type information was available, assuming
```

Packet Distribution Tree

In this option, we take the traffic form a packet and then drive it through the “http,tree” option under the “-z” parameter to count the number of the HTTP requests, their mods as well as the status code. This is a rather modular approach that is very easy to understand and analyse. Here in our case, we took the packet that we captured earlier and then drove it through the tree option that gave us the Information that a total of 126 requests were generated out of which 14 gave back the “200 OK”. It means that the rest of them either gave back an error or were redirected to another server giving back a 3XX series status code.

```
tshark -r wlan.pcap -z http,tree -q
```

```
root@kali:~# tshark -r wlan.pcap -z http,tree -q
Running as user "root" and group "root". This could be dangerous.

================================================================================
HTTP/Packet Counter:
Topic / Item                  Count         Average       Min val       Max val       Rate (ms)     Percent
--------------------------------------------------------------------------------
Total HTTP Packets            248                                                   0.0038        100%
 HTTP Request Packets         126                                                   0.0019        50.81%
  GET                         126                                                   0.0019        100.00%
 HTTP Response Packets        122                                                   0.0019        49.19%
  3xx: Redirection            105                                                   0.0016        86.07%
   304 Not Modified           101                                                   0.0015        96.19%
   302 Found                  3                                                     0.0000        2.86%
   301 Moved Permanently      1                                                     0.0000        0.95%
  2xx: Success                17                                                    0.0003        13.93%
   200 OK                     14                                                    0.0002        82.35%
   204 No Content             3                                                     0.0000        17.65%
  ???: broken                 0                                                     0.0000        0.00%
  5xx: Server Error           0                                                     0.0000        0.00%
  4xx: Client Error           0                                                     0.0000        0.00%
  1xx: Informational          0                                                     0.0000        0.00%
 Other HTTP Packets           0                                                     0.0000        0.00%
```

Packet Length Tree

As long as we are talking about the Tree option, let's explore it a bit. We have a large variety of ways in which we can use the tree option in combination with other option. To demonstrate that, we decided to use the packet length option with the tree option. This will sort the data on the basis of the size of the packets and then generate a table with it. Now, this table will not only consist of the length of the packets, but it will also have the count of the packet. The minimum value of the length in the range of the size of the packets. It will also calculate the size as well as the Percentage of the packets inside the range of packet length

```
tshark -r wlan.pcap -z plen,tree -q
```

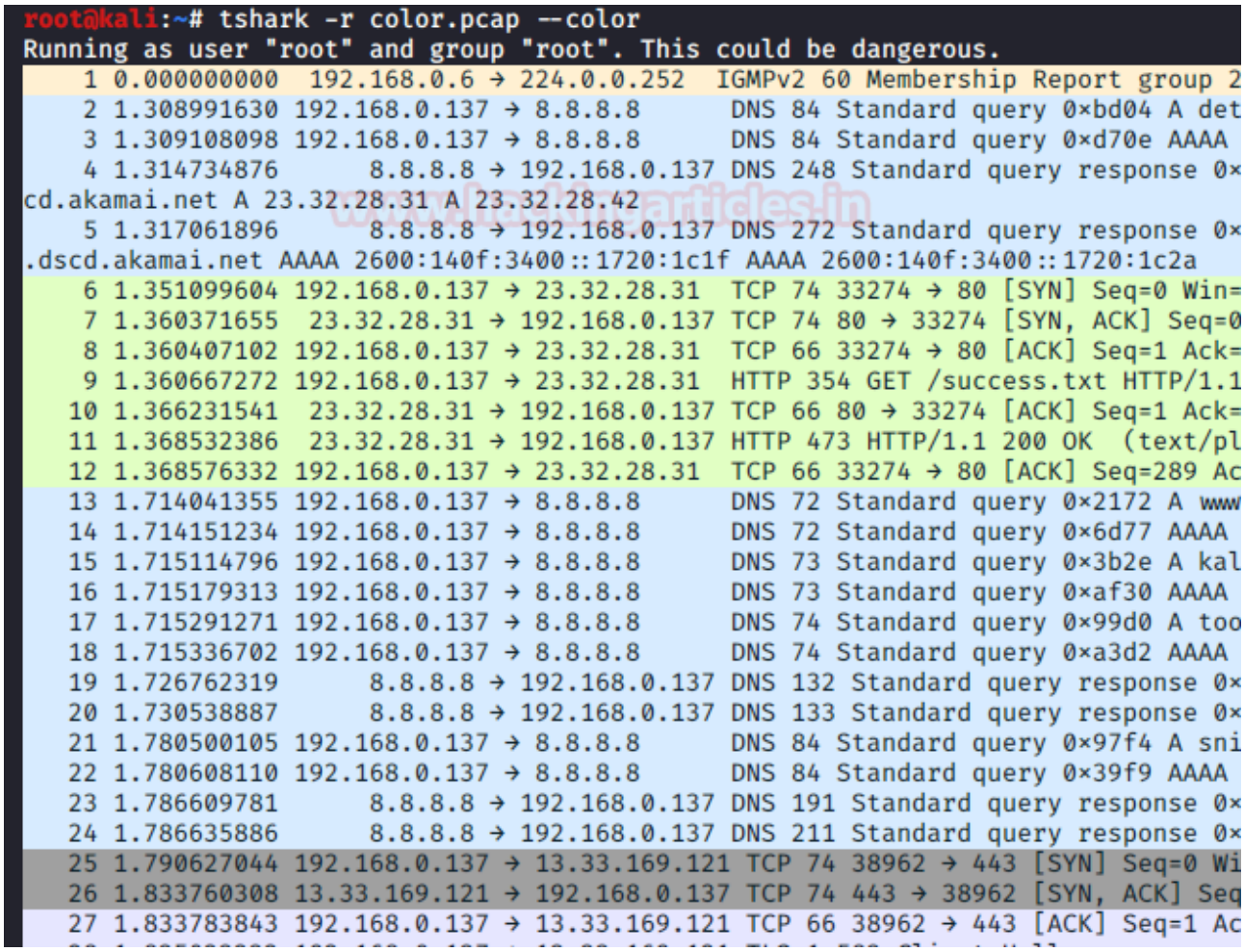
```
root@kali:~# tshark -r wlan.pcap -z plen,tree -q
Running as user "root" and group "root". This could be dangerous.

================================================================================
Packet Lengths:
Topic / Item          Count         Average       Min val       Max val       Rate (ms)     Percent
--------------------------------------------------------------------------------
Packet Lengths        66690         225.14        50            1582          0.0004        100%
 0-19                 0             -             -             -             0.0000        0.00%
 20-39                0             -             -             -             0.0000        0.00%
 40-79                42235         54.68         50            76            0.0003        63.33%
 80-159               9477          134.43        86            159           0.0001        14.21%
 160-319              6071          257.61        160           317           0.0000        9.10%
 320-639              2724          390.89        320           639           0.0000        4.08%
 640-1279             456           844.38        640           1278          0.0000        0.68%
 1280-2559            5727          1469.77       1280          1582          0.0000        8.59%
 2560-5119            0             -             -             -             0.0000        0.00%
 5120 and greater     0             -             -             -             0.0000        0.00%
```

Color Based Output Analysis

Note: Your terminal must support color output in order for this option to work correctly.

We can enable the coloring of packets according to standard Wireshark color filters. On Windows, colors are limited to the standard console character attribute colors. In this option, we can set up the colors according to the display filter. This helps in quickly locating a specific packet in the bunch of similar packets. It also helps in locating Handshakes in communication traffic. This can be enabled using the following command.

```
tshark -r color.pcap --color
```



Ring Buffer Analysis

By default, the TShark to runs in the “multiple files” mode. In this mode, the TShark writes into several capture files. When the first capture file fills up to a certain capacity, the TShark switches to the next file and so on. The file names that we want to create can be stated using the -w parameter. The number of files, creation data and creation time will be concatenated with the name provided next to -w parameter to form the complete name of the file.

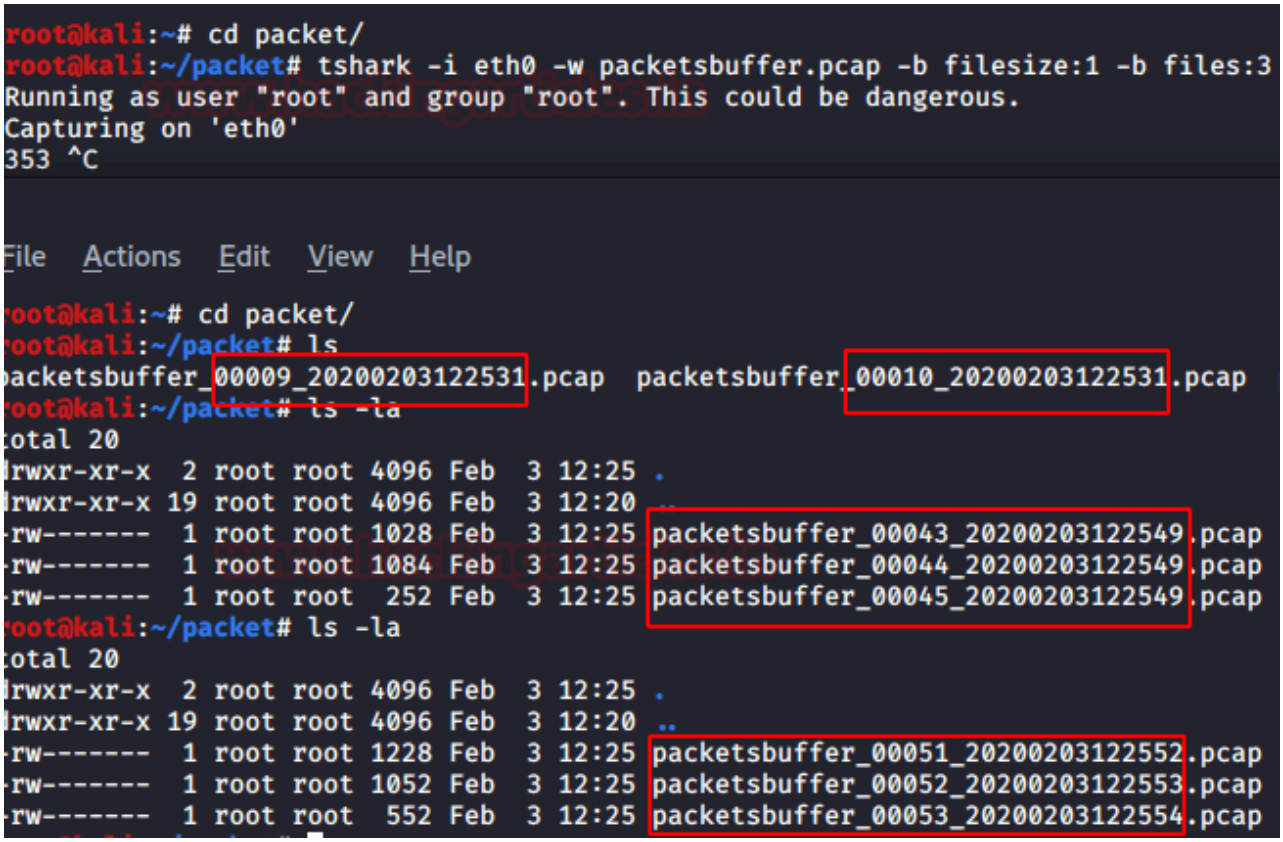
The files option will fill up new files until the number of files is specified. at that moment the TShark will discard data in the first file and start writing to that file and so on. If the files option is not set, new files filled up until one of the captures stops conditions matches or until the disk is full.

There are a lot of criteria upon which the ring buffer works but, in our demonstration, we used 2 of them. Files and the Filesize.

files: value begin again with the first file after value number of files were written (form a ring buffer). This value must be less than 100000.

filesize: value switches to the next file after it reaches a size of value kB. Note that the file size is limited to a maximum value of 2 GiB.

```
tshark -I eth0 -w packetsbuffer.pcap -b filesize:1 -b file:3
```



Auto-Stop

Under the huge array of the options, we have one option called auto-stop. As the name tells us that it will stop the traffic capture after the criteria are matched.

Duration

We have a couple of options, in our demonstration, we used the duration criteria. We specified the duration to 10. This value is in seconds. So, the capture tells us that in the time of 10 seconds, we captured 9 packets.

```
tshark -i eth0 -a duration:10
```

```
root@kali:~# tshark -i eth0 -a duration:10
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
    1 0.000000000 192.168.0.137 → 52.73.26.88  TLSv1.2 841 Application Data
    2 0.228521540  52.73.26.88 → 192.168.0.137 TLSv1.2 191 Application Data
    3 0.228547939 192.168.0.137 → 52.73.26.88  TCP 66 50990 → 443 [ACK] Seq
    4 0.228611423  52.73.26.88 → 192.168.0.137 TCP 66 443 → 50990 [ACK] Seq
    5 0.228615310 192.168.0.137 → 52.73.26.88  TCP 66 [TCP Dup ACK 3#1] 509
    6 4.977273190 192.168.0.137 → 13.249.226.169 TCP 66 34476 → 443 [ACK] S
    7 5.010827964 13.249.226.169 → 192.168.0.137 TCP 66 [TCP ACKed unseen s
    8 6.512880036 192.168.0.137 → 104.79.123.250 TCP 66 37426 → 443 [ACK] S
    9 6.623442093 104.79.123.250 → 192.168.0.137 TCP 66 [TCP ACKed unseen s
9 packets captured
```

File Size

Now another criterion for the auto-stop option is the file size. The TShark will stop writing to the specified capture file after it reaches a size provided by the user. In our demonstration, we set the filesize to 1. This value is in kB. We used the directory listing command to show that the capture was terminated as soon as the file reached the size of 1 kB.

```
tshark -i eth0 -w 1.pcap -a filesize:1
```

```
root@kali:~# tshark -i eth0 -w 1.pcap  -a filesize:1
Running as user "root" and group "root". This could be dange
Capturing on 'eth0'
6
root@kali:~# ls -la
total 16156
drwxr-xr-x 19 root root     4096 Feb  3 12:33 .
drwxr-xr-x 18 root root     4096 Nov 25 12:38 ..
-rw-------  1 root root     1172 Feb  3 12:33 1.pcap
-rw-r--r--  1 root root     5161 Feb  3 12:24 .bash_history
```

Data-Link Types

At last, we can also modify the statistics of the captured traffic data based on the Data-Link Types. For that we will have to use an independent parameter, “-L”. In our demonstration, we used the “-L” parameter to show that we have data links like EN10MB specified for the Ethernet Traffic and others.

```
tshark -L
```

```
root@kali:~/packet# tshark -L
Running as user "root" and group "root". This could be da
Data link types of interface eth0 (use option -y to set)
  EN10MB (Ethernet)
  DOCSIS (DOCSIS)
```