??????? ???????????

- Руководство по TShark

?????????? ?? TShark



В этой статье пойдет речь о TShark, который является популярным анализатором сетевых протоколов. Он позволяет пользователю захватывать пакеты данных из сети. Инструмент также дает возможность читать или анализировать ранее захваченные пакеты данных из сохраненных файлов.

??????? ??????

Как известно, ***сетевой трафик*** или трафик данных — это объем данных, передаваемых по сети за определенный отрезок времени. Сетевые данные в компьютерных сетях представлены в виде пакетов сетевых данных. Анализ этих сетевых пакетов обеспечивает безопасность сети, поскольку помогает пользователю отслеживать свой трафик. Если сеть отличается каким-то подозрительным объемом трафика данных, который является возможным признаком атаки, то TShark может помочь пользователю узнать об этом до того, как станет слишком поздно. Атака также будет остановлена, поскольку отчеты о трафике данных дают представление обо всей картине в целом.

Объем трафика — это термин, который используется при анализе сетевого трафика. Объем сетевого трафика — это показатель общей проделанной работы. Он определяется как средняя интенсивность трафика данных и период времени исследования его пакета сетевых данных.

?????????? ? TShark

Tshark, популярный и мощный инструмент командной строки, он используется в качестве сетевого анализатора. Инструмент был разработан компанией Wireshark. Его рабочая система очень похожа на Tcpdump, но у него есть несколько мощных декодеров и фильтров. TShark способен захватывать информацию о пакетах данных различных сетевых уровней и отображать их в вариативных форматах.

TShark применяется для ***анализа сетевого трафика в реальном времени***, и он способен читать файлы формата .pcap для анализа информации. Подобный анализ этих соединений помогает специалистам по безопасности выявить имеющуюся сетевую проблему.

TShark — это инструмент командной строки, который способен делать все, что делает и Wireshark. Итак, стоит начать процесс знакомства с TShark. Пользователь запустит этот инструмент и изучит его возможности. Чтобы проверить все параметры, необходимо использовать следующую команду:

```
tshark -h
```



```
root@kali:~# tshark -h
Running as user "root" and group "root". This could be dangerous.
TShark (Wireshark) 3.0.5 (Git v3.0.5 packaged as 3.0.5-1)
Dump and analyze network traffic.
See https://www.wireshark.org for more information.

Usage: tshark [options] ...

Capture interface:
  -i <interface>           name or idx of interface (def: first non-loopback)
  -f <capture filter>      packet filter in libpcap filter syntax
  -s <snaplen>             packet snapshot length (def: appropriate maximum)
  -p                       don't capture in promiscuous mode
  -I                       capture in monitor mode, if available
  -B <buffer size>         size of kernel buffer (def: 2MB)
  -y <link type>           link layer type (def: first appropriate)
  --time-stamp-type <type> timestamp method for interface
  -D                       print list of interfaces and exit
  -L                       print list of link-layer types of iface and exit
  --list-time-stamp-types  print list of timestamp types for iface and exit

Capture stop conditions:
  -c <packet count>        stop after n packets (def: infinite)
  -a <autostop cond.> ...  duration:NUM - stop after NUM seconds
                           filesize:NUM - stop this file after NUM KB
                              files:NUM - stop after NUM files
Capture output:
  -b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                           interval:NUM - create time intervals of NUM secs
                           filesize:NUM - switch to next file after NUM KB
                              files:NUM - ringbuffer: replace after NUM files
Input file:
  -r <infile⟼              set the filename to read from (or '-' for stdin)

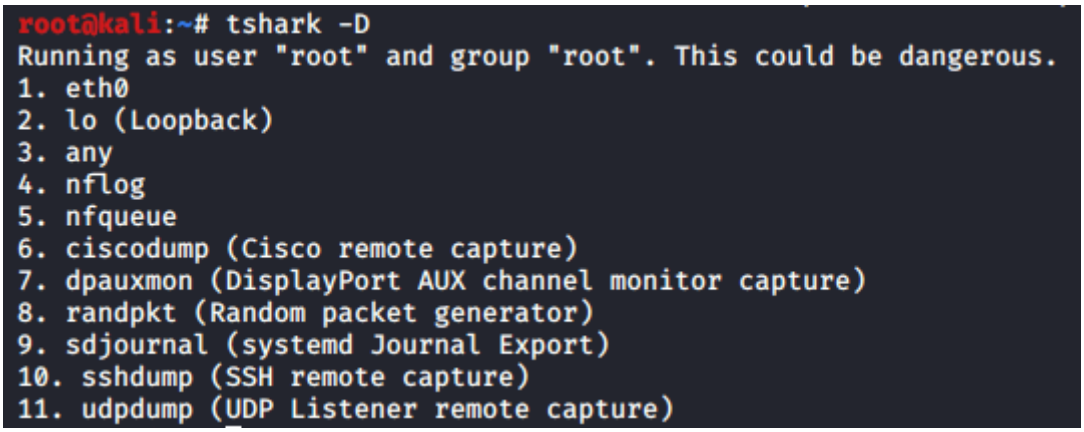
Processing:
  -2                       perform a two-pass analysis
  -M <packet count>        perform session auto reset
  -R <read filter>         packet Read filter in Wireshark display filter syntax
                           (requires -2)
  -Y <display filter>      packet displaY filter in Wireshark display filter
                           syntax
  -n                       disable all name resolutions (def: all enabled)
  -N <name resolve flags>  enable specific name resolution(s): "mnNtdv"
  -d <layer_type>==<selector>,<decode_as_protocol> ...
                           "Decode As", see the man page for details
                           Example: tcp.port==8888,http
  -H <hosts file>          read a list of entries from a hosts file, which will
```

?????? ???????????

TShark показывает список интерфейсов, трафик которых он может перехватывать. Каждый интерфейс называется его серийным номером, и, как читатели могут увидеть, за ним следует текстовое описание сетевого интерфейса. Эти интерфейсы могут быть заданы с помощью параметра «**-i**», который применяется для указания сети, трафик которой пользователь хочет захватить. Чтобы проверить все интерфейсы, человек должен

использовать другой параметр «**-d**», как показано на рисунке ниже:

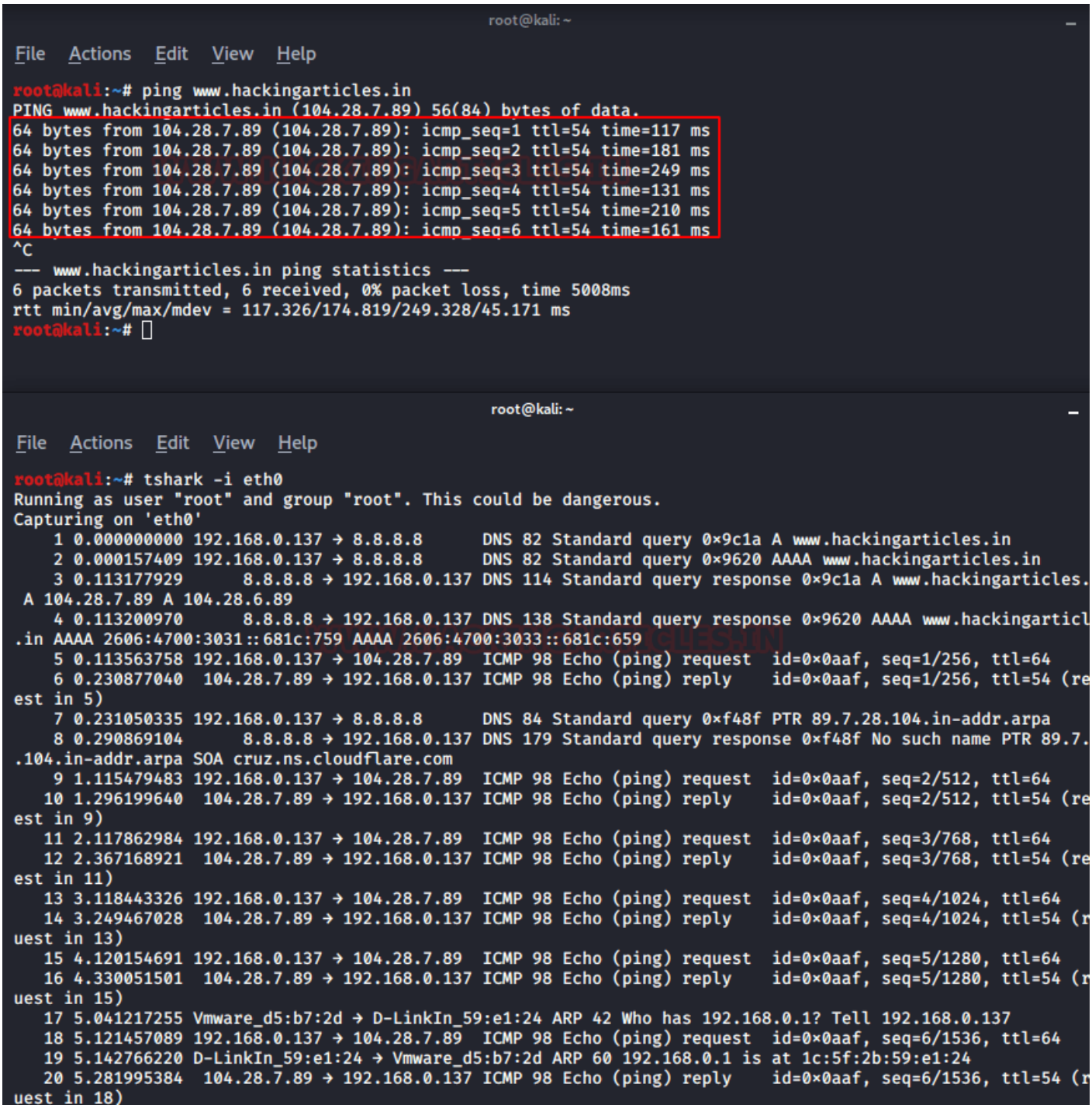
```
tshark -D
```



?????? ???????

Настала пора попробовать захватить трафик. У пользователя есть различные возможности для выбора интерфейса для захвата трафика, поэтому можно выбрать любой из них в зависимости от своих потребностей. Но в данном сценарии интерфейс, который пользователь собирается исследовать, — это «**eth0**». Для того чтобы захватить трафик, нужно инициировать его тоже, так как человек тестирует это в контролируемой сети. Для этого он будет использовать команду ***ping***, а затем просто укажет имя интерфейса с помощью параметра «**-i**», как показано на рисунке ниже:

```
ping www.hackingarticles.in
tshark -i eth0
```



Как видно на картинке, инструмент выполняет трехстороннее рукопожатие, а затем запускает процесс запроса ICMP и получает ответ.

Promiscuous mode

В сети этот режим используется в качестве контроллера интерфейса, который заставляет TShark передавать весь трафик, что он получает, на центральный процессор, а не передавать его в другое место, где происходит снифинг пакетов. Такое иметь место быть на маршрутизаторе или на компьютере, подключенном к проводной локальной сети.

При использовании Promiscuous mode нужно будет настроить его с помощью **ifconfig** так, чтобы инструмент давал возможность пользователю захватывать пакеты данных всей сети. Поэтому человек начинает с пинга веб-сайта и пытается захватить его пакеты данных.

```
victim@ubuntu:~$ ping www.hackingarticles.in
PING www.hackingarticles.in (104.28.7.89) 56(84) bytes of data.
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=1 ttl=54 time=194 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=2 ttl=54 time=244 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=3 ttl=54 time=301 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=4 ttl=54 time=235 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=5 ttl=54 time=196 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=6 ttl=54 time=116 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=7 ttl=54 time=183 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=8 ttl=54 time=217 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=9 ttl=54 time=149 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=10 ttl=54 time=201 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=11 ttl=54 time=177 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=12 ttl=54 time=355 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=13 ttl=54 time=245 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=14 ttl=54 time=305 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=15 ttl=54 time=153 ms
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=16 ttl=54 time=254 ms
```

Теперь следует настроить этот режим. Пользователь будет выполнять эти команды и попытается захватить пакеты:

```
ifconfig eth0 promisc
tshark -i eth0
```

```
root@kali:~# ifconfig eth0 promisc
root@kali:~# tshark -i eth0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
    1 0.000000000  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=95/2432
    2 0.136847861  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0×1b86, seq=95/2432
    3 0.767140702 fe80::8dcf:3961:7c07:7841 → ff02::fb     MDNS 180 Standard query 0×0000 PTR _ft
._tcp.local, "QM" question PTR _sftp-ssh._tcp.local, "QM" question PTR _webdavs._tcp.local, "QM"
    4 0.767158404  192.168.0.6 → 224.0.0.251  MDNS 160 Standard query 0×0000 PTR _ftp._tcp.local,
"QM" question PTR _sftp-ssh._tcp.local, "QM" question PTR _webdavs._tcp.local, "QM" question PTR
    5 0.767231655 fe80::20c:29ff:fed5:b72d → ff02::1:ff00:0 ICMPv6 86 Neighbor Solicitation for :
    6 1.001500570  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=96/2457
    7 1.232830355  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0×1b86, seq=96/2457
    8 2.002672796  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=97/2483
    9 2.534998232  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0×1b86, seq=97/2483
   10 3.003729111  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=98/2508
   11 3.151781403  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0×1b86, seq=98/2508
   12 4.005684733  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=99/2534
   13 4.221686910  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0×1b86, seq=99/2534
   14 4.366369429 OneplusT_55:6d:66 → Broadcast    ARP 60 Who has 192.168.0.1? Tell 192.168.0.7
^C   15 5.006460197  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0×1b86, seq=100/2
```

?????????? ???????

Tshark обладает удивительными функциями, с которыми пользователь сможет работать более эффективно. Он будет способен получить доступ к этим функциям, используя различные параметры. Одним из таких параметров является «**-c**», он позволяет человеку захватить точное количество данных, которые ему нужны, и будет отображать только их. Эта опция помогает уточнить результат захваченного трафика.

```
tshark -i eth0 -c 10
```

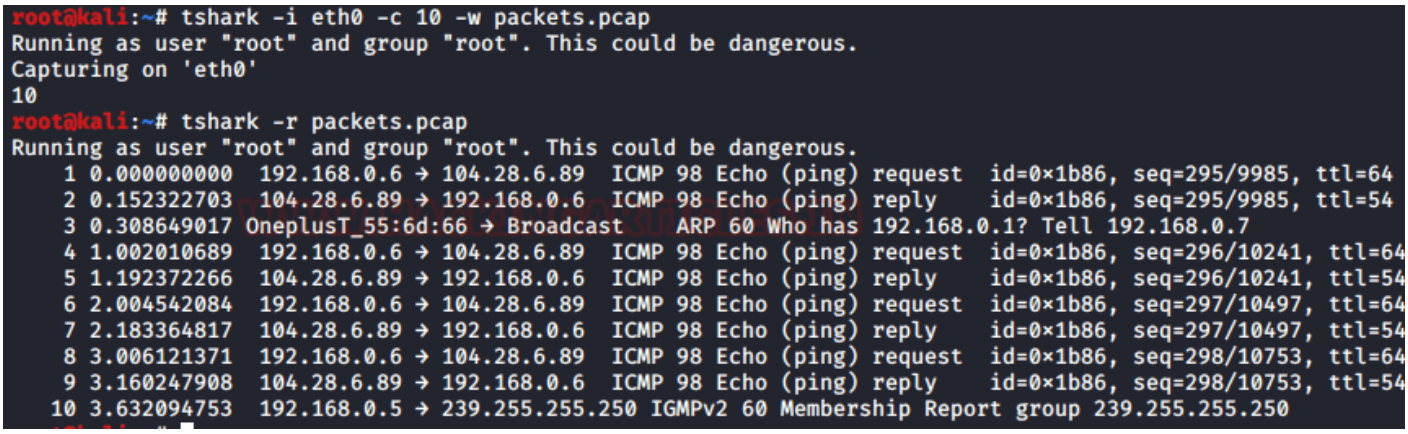
?????? ? ??????????????? ?????

В Tshark пользователь может редактировать и читать файл формата .**pcap**. Опция записи (**-w**) позволяет записывать исходные пакетные данные в стандартный формат .pcap, в то время как опция чтения (**-r**) дает возможность считывать эти необработанные выходные пакеты данных в желаемом порядке. Для записи пакетов в файл формата .pcap следует использовать следующую команду:

```
tshark -i eth0 -c 10 -w packets.pcap
```

Чтобы прочитать содержимое файла, нужно ввести следующую команду:

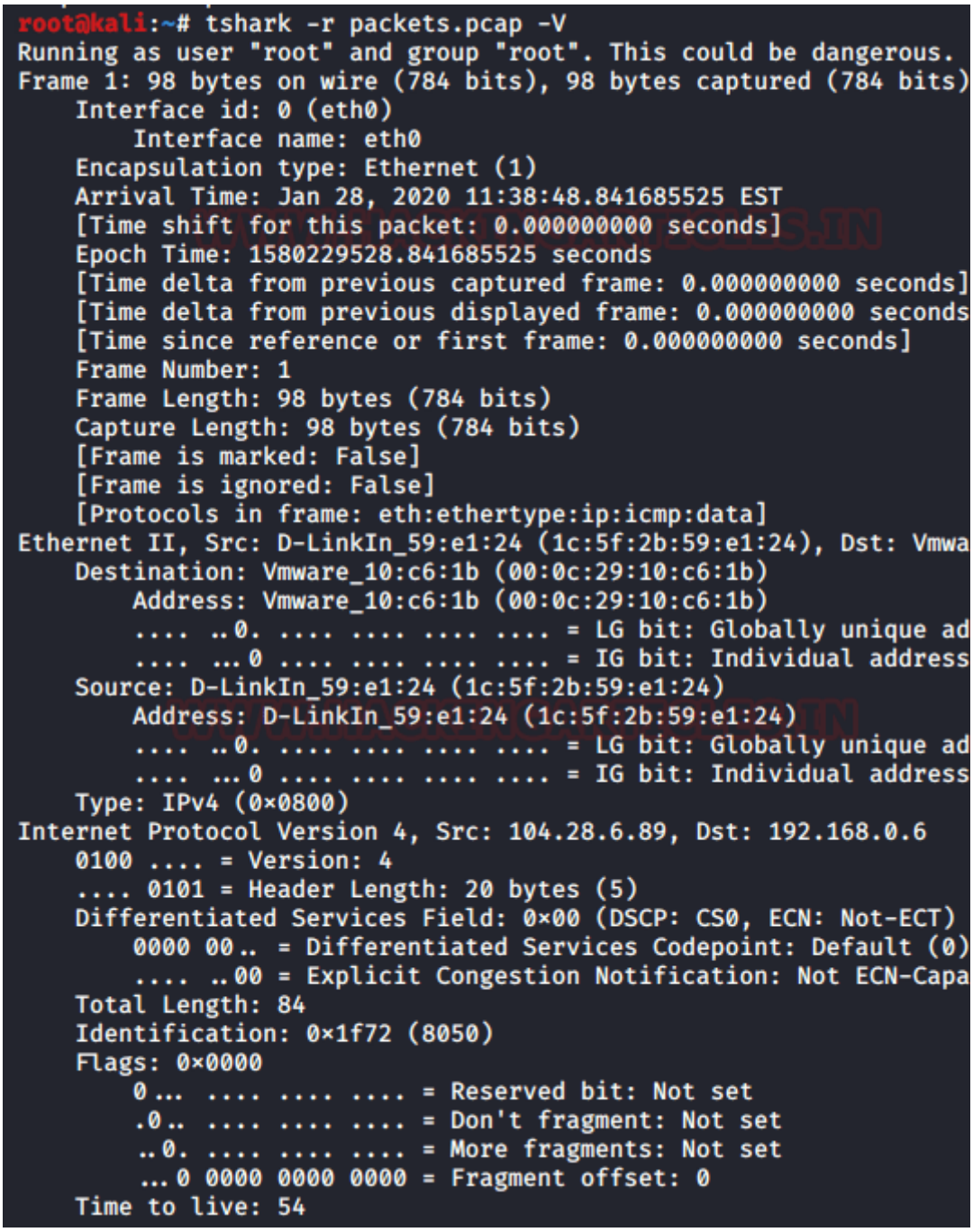
```
tshark -r packets.pcap
```



Verbose mode

Verbose mode дает пользователю дополнительные сведения о пакетах в трафике. Используя его, человек может видеть информацию, содержащуюся в каждом пакете. Для этой опции следует применить параметр «**–v**».

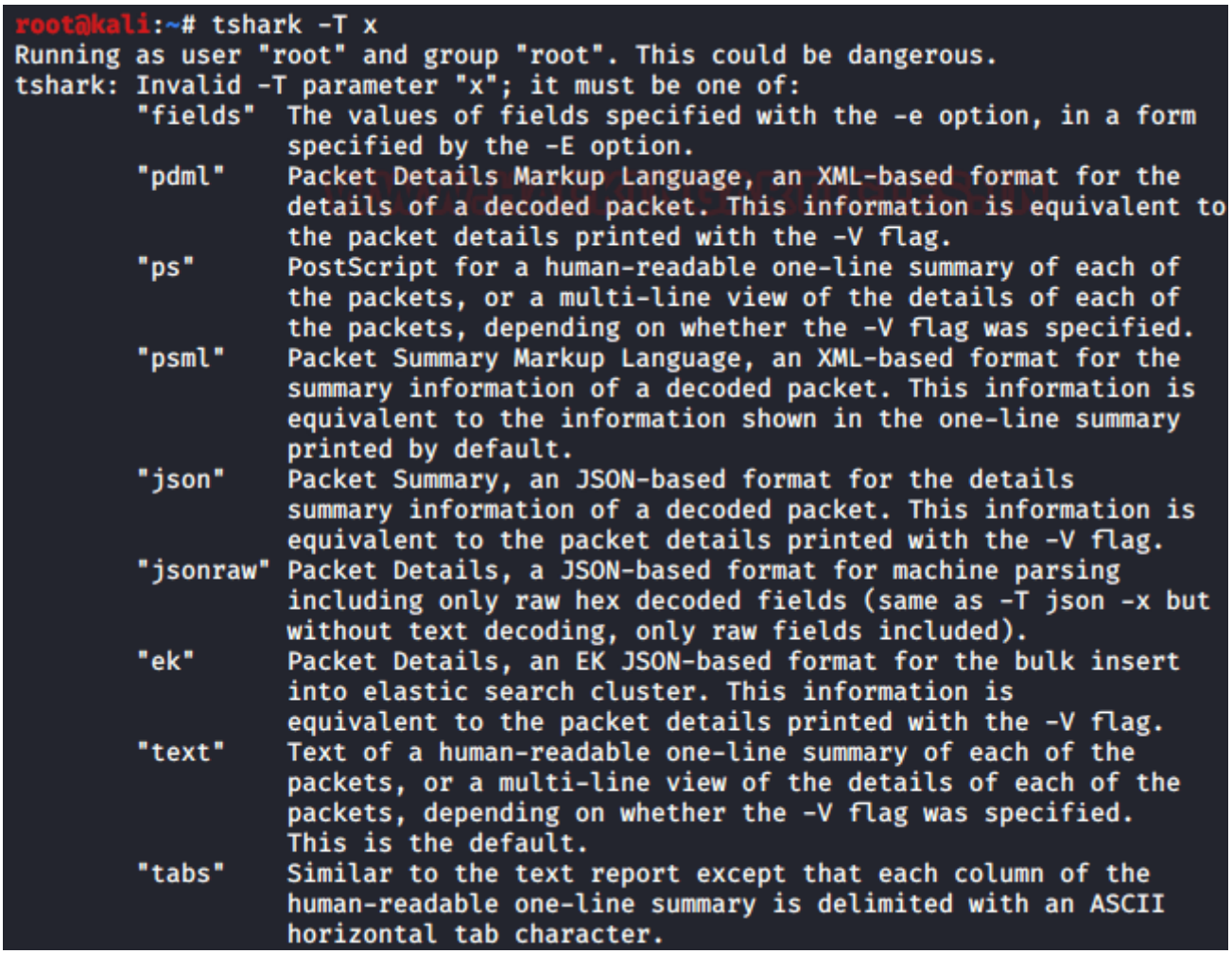
```
tshark -r packets.pcap -V
```



?????? ???????? ??????

Для удобства в TShark есть опция «**-t**», которая позволяет сохранять декодированные пакеты в различных выходных форматах. Человек может установить формат выходных данных таким образом, чтобы его было легко прочесть. Чтобы просмотреть все доступные параметры, нужно ввести следующую команду:

```
tshark -T x
```



PDML

PDML расшифровывается как ***Packet Details Mark-Up Language***, который основан на XML. Эта информация вполне эквивалентна режиму Verbose mode, о котором говорилось ранее. А чтобы получить вывод данных в этом формате, надо ввести следующую команду:

```
tshark -r packets.pcap -T pdml
```

```
root@kali:~# tshark -r packets.pcap -T pdml
Running as user "root" and group "root". This could be dangerous.
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="pdml2html.xsl"?>
<!-- You can find pdml2html.xsl in /usr/share/wireshark or at https://code.wireshark.org/
w/gitweb?p=wireshark.git;a=blob_plain;f=pdml2html.xsl. -->
<pdml version="0" creator="wireshark/3.0.5" time="Tue Jan 28 11:46:48 2020" capture_file=
ets.pcap">
<packet>
  <proto name="geninfo" pos="0" showname="General information" size="98">
    <field name="num" pos="0" show="1" showname="Number" value="1" size="98" />
    <field name="len" pos="0" show="98" showname="Frame Length" value="62" size="98" />
    <field name="caplen" pos="0" show="98" showname="Captured Length" value="62" size="98
    <field name="timestamp" pos="0" show="Jan 28, 2020 11:46:40.459901028 EST" showname="
red Time" value="1580230000.459901028" size="98" />
  </proto>
  <proto name="frame" showname="Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (
its) on interface 0" size="98" pos="0">
    <field name="frame.interface_id" showname="Interface id: 0 (eth0)" size="0" pos="0" s
0">
      <field name="frame.interface_name" showname="Interface name: eth0" size="0" pos="0"
="eth0" />
    </field>
    <field name="frame.encap_type" showname="Encapsulation type: Ethernet (1)" size="0" p
" show="1" />
    <field name="frame.time" showname="Arrival Time: Jan 28, 2020 11:46:40.459901028 EST"
="0" pos="0" show="Jan 28, 2020 11:46:40.459901028 EST" />
    <field name="frame.offset_shift" showname="Time shift for this packet: 0.000000000 se
" size="0" pos="0" show="0.000000000" />
    <field name="frame.time_epoch" showname="Epoch Time: 1580230000.459901028 seconds" si
" pos="0" show="1580230000.459901028" />
```

PS

PS расшифровывается как ***PostScript***. Эти выходные данные представлены в виде однострочной сводки по каждому пакету данных или многострочного подробного представления пакета данных в зависимости от спецификации. Эти однострочники очень быстро читаются, а так они вполне надежны. Для вывода данных в этом формате следует использовать следующую команду:

```
tshark -r packets.pcap -T ps
```

```
root@kali:~# tshark -r packets.pcap -T ps
Running as user "root" and group "root". This could be dangerous.
%!
%!PS-Adobe-2.0
%
% Wireshark - Network traffic analyzer
% By Gerald Combs <gerald@wireshark.org>
% Copyright 1998 Gerald Combs
%
%%Creator: Wireshark
%%Title: Wireshark output
%%DocumentFonts: Helvetica Monaco
%%EndComments
%!

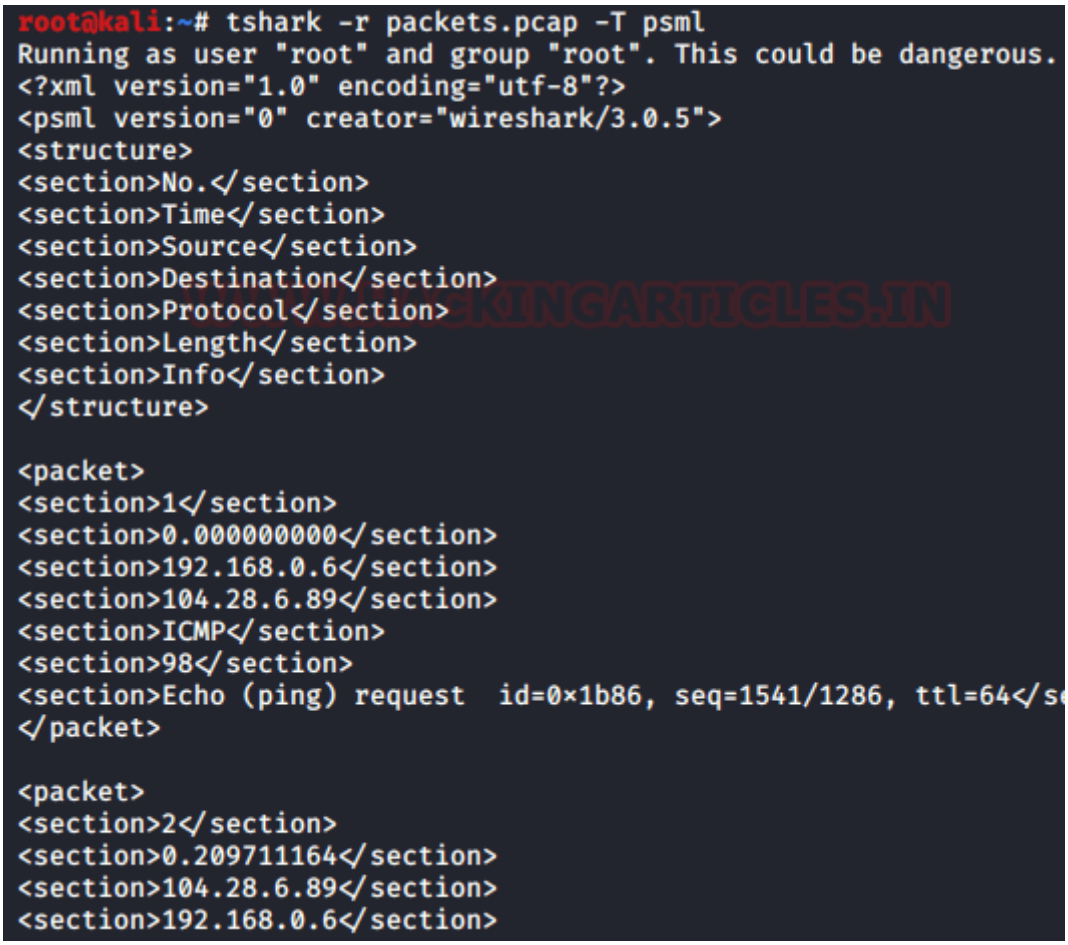
%
% Ghostscript http://ghostscript.com/ can convert postscript to pdf files.
%
% To convert this postscript file to pdf, type (for US letter format):
% ps2pdf filename.ps
%
% or (for A4 format):
% ps2pdf -sPAPERSIZE=a4 filename.ps
%
%  ...  and of course replace filename.ps by your current filename.
%
% The pdfmark's below will help converting to a pdf file, and have no
% effect when printing the postscript directly.
%
```

WWW.HACKINGARTICLES.IN

PSML

PSML расшифровывается как ***Packet Summary Mark-Up Language***. Этот формат на основе XML, такой как PDML, он суммирует подробную информацию о пакетах. И для получения этого типа формата данных надо ввести следующее:

```
tshark -r packets.pcap -T psml
```



JSON

JSON расшифровывается как ***Java-Script Object Notation***. Это открытый стандартный формат файла, который отображает текст в удобно читаемой форме. Информация в этом формате полностью документирована и передана в **Wolfram**. Чтобы увидеть, что представляют собой пакеты данных в этом формате, необходимо ввести:

```
tshark -r packets.pcap -T json
```



EK

Это функция формата **JSON** с разделителями новой строки для массового импорта с опцией гибкого поиска. Для этого формата следует использовать следующую команду:

```
tshark -r packets.pcap -T ek
```

```
root@kali:~# tshark -r packets.pcap -T ek
Running as user "root" and group "root". This could be dangerous.
{"index":{"_index":"packets-2020-01-28","_type":"pcap_file"}}
{"timestamp":"1580230675786","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_i
type":"1","frame_frame_time":"Jan 28, 2020 11:57:55.786675361 EST","frame_frame_offset_shift":"0.
6675361","frame_frame_time_delta":"0.000000000","frame_frame_time_delta_displayed":"0.000000000",
rame_number":"1","frame_frame_len":"98","frame_frame_cap_len":"98","frame_frame_marked":"0","fram
ethertype:ip:icmp:data"},"eth":{"eth_eth_dst":"1c:5f:2b:59:e1:24","eth_dst_eth_dst_resolved":"D-L
:24","eth_dst_eth_addr_resolved":"D-LinkIn_59:e1:24","eth_dst_eth_lg":"0","eth_dst_eth_ig":"0","e
esolved":"Vmware_10:c6:1b","eth_src_eth_addr":"00:0c:29:10:c6:1b","eth_src_eth_addr_resolved":"Vm
g":"0","eth_eth_type":"0x00000800"},"ip":{"ip_ip_version":"4","ip_ip_hdr_len":"20","ip_ip_dsfield
p_dsfield_ip_dsfield_ecn":"0","ip_ip_len":"84","ip_ip_id":"0x00003c1c","ip_ip_flags":"0x00004000"
:"1","ip_flags_ip_flags_mf":"0","ip_flags_ip_frag_offset":"0","ip_ip_ttl":"64","ip_ip_proto":"1",
us":"2","ip_ip_src":"192.168.0.6","ip_ip_addr":["192.168.0.6","104.28.6.89"],"ip_ip_src_host":"19
9"],"ip_ip_dst":"104.28.6.89","ip_ip_dst_host":"104.28.6.89"},"icmp":{"icmp_icmp_type":"8","icmp_
icmp_icmp_checksum_status":"1","icmp_icmp_ident":["7046","34331"],"icmp_icmp_seq":"1541","icmp_ic
020 11:57:55.000000000 EST","icmp_icmp_data_time_relative":"0.786675361","icmp_data":{"data_data_
6:17:18:19:1a:1b:1c:1d:1e:1f:20:21:22:23:24:25:26:27:28:29:2a:2b:2c:2d:2e:2f:30:31:32:33:34:35:36
{"index":{"_index":"packets-2020-01-28","_type":"pcap_file"}}
{"timestamp":"1580230675996","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_i
type":"1","frame_frame_time":"Jan 28, 2020 11:57:55.996386525 EST","frame_frame_offset_shift":"0.
```

?????

Текст представляет собой понятную для человека однострочную сводку каждого из пакетов. Это самый простой из форматов. Для него человек будет использовать следующую команду:

```
tshark -r packets.pcap -T text
```

```
root@kali:~# tshark -r packets.pcap -T text
Running as user "root" and group "root". This could be dangerous.
    1 0.000000000  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0x1b86, seq=1541/12
86, ttl=64
    2 0.209711164  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0x1b86, seq=1541/12
86, ttl=54 (request in 1)
    3 1.001906657  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0x1b86, seq=1542/15
42, ttl=64
    4 1.192770973  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0x1b86, seq=1542/15
42, ttl=54 (request in 3)
    5 2.003365632  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0x1b86, seq=1543/17
98, ttl=64
    6 2.434560259  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0x1b86, seq=1543/17
98, ttl=54 (request in 5)
    7 3.003769942  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0x1b86, seq=1544/20
54, ttl=64
    8 3.347729784  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0x1b86, seq=1544/20
54, ttl=54 (request in 7)
    9 4.003967430  192.168.0.6 → 104.28.6.89  ICMP 98 Echo (ping) request  id=0x1b86, seq=1545/23
10, ttl=64
   10 4.163455725  104.28.6.89 → 192.168.0.6  ICMP 98 Echo (ping) reply    id=0x1b86, seq=1545/23
10, ttl=54 (request in 9)
```

???????

Этот формат очень похож на текст, за исключением того, что он включает в себя горизонтальное разделение ASCII (oxo9) между каждым столбцом. Чтобы попробовать этот формат, следует ввести:

```
tshark -r packets.pcap -T tabs
```

```
root@kali:~# tshark -r packets.pcap -T tabs
Running as user "root" and group "root". This could be dangerous.
    1   0.000000000      192.168.0.6    →       104.28.6.89     ICMP    98      Echo (ping) reque
st  id=0×1b86, seq=1541/1286, ttl=64
    2   0.209711164      104.28.6.89    →       192.168.0.6     ICMP    98      Echo (ping) reply
    id=0×1b86, seq=1541/1286, ttl=54 (request in 1)
    3   1.001906657      192.168.0.6    →       104.28.6.89     ICMP    98      Echo (ping) reque
st  id=0×1b86, seq=1542/1542, ttl=64
    4   1.192770973      104.28.6.89    →       192.168.0.6     ICMP    98      Echo (ping) reply
    id=0×1b86, seq=1542/1542, ttl=54 (request in 3)
    5   2.003365632      192.168.0.6    →       104.28.6.89     ICMP    98      Echo (ping) reque
st  id=0×1b86, seq=1543/1798, ttl=64
    6   2.434560259      104.28.6.89    →       192.168.0.6     ICMP    98      Echo (ping) reply
    id=0×1b86, seq=1543/1798, ttl=54 (request in 5)
    7   3.003769942      192.168.0.6    →       104.28.6.89     ICMP    98      Echo (ping) reque
st  id=0×1b86, seq=1544/2054, ttl=64
    8   3.347729784      104.28.6.89    →       192.168.0.6     ICMP    98      Echo (ping) reply
    id=0×1b86, seq=1544/2054, ttl=54 (request in 7)
    9   4.003967430      192.168.0.6    →       104.28.6.89     ICMP    98      Echo (ping) reque
st  id=0×1b86, seq=1545/2310, ttl=64
   10   4.163455725      104.28.6.89    →       192.168.0.6     ICMP    98      Echo (ping) reply
    id=0×1b86, seq=1545/2310, ttl=54 (request in 9)
```

??????? ????? ??????????????? ???????? ? ????????????? ????????

Когда человек пытается записать пакеты данных в реальном времени в файл формата .pcap, он сжимает все данные в более мелкие сегменты. Чтобы лучше понять эти пакеты данных, нужно их декодировать, что приведет к разнице в размере файла. Чтобы проверить размер любого файла в данный момент, нужно использовать следующую команду:

```
ls -lh packets.p*
```

```
root@kali:~# ls -lh packets.p*
-rw------- 1 root root 624 Jan 28 11:46 packets.pcap
-rw-r--r-- 1 root root 21K Jan 28 11:49 packets.pdml
```

Как уже было сказано, есть большая разница в этих файлах, поэтому человек использует методы декодирования для извлечения нужной ему информации.

????????????? PDML-????? HTML-????????

Единственное различие между Wireshark и TShark заключается в том, что Wireshark — это инструмент на основе графического интерфейса, а TShark – инструмент на основе командной строки. Но с помощью некоторого внешнего источника человек также может просматривать пакеты данных в формате HTML. Поэтому для достижения этой цели сначала

нужно сохранить пакеты данных в формате PDML, а затем преобразовать их в XML-файл с помощью следующей команды:

```
tshark -r packets.pcap -T pdml > packets.xml
```

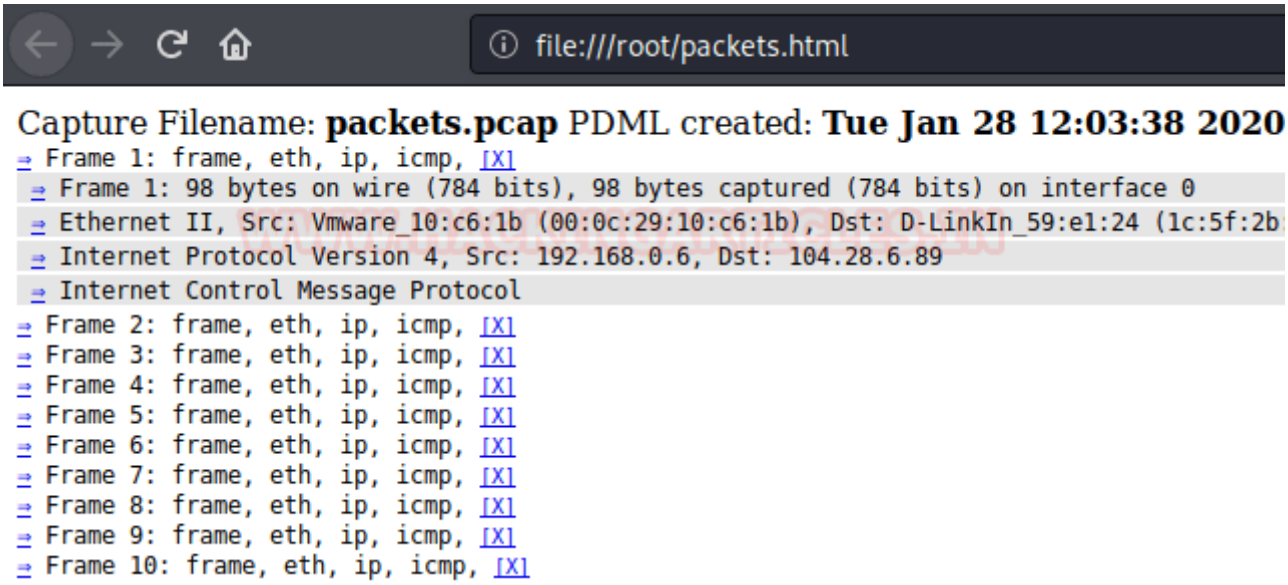
XML-файл будет сохранен в папке **/usr/share/wireshark/pdml2html.xsl**. Итак, человек собирается использовать инструмент ***xsltproc*** для запуска этого файла, который поможет создать HTML-страницу. Создание HTML-страницы приведет к форматированию всей ненужной информации и позволит просматривать только полезные данные. Для создания HTML-страницы, следует использовать следующую команду:

```
xsltproc /usr/share/wireshark/pdml2html.xsl packets.xml > packets.html
```

```
root@kali:~# tshark -r packets.pcap -T pdml > packets.xml
Running as user "root" and group "root". This could be dangerous.
root@kali:~# xsltproc /usr/share/wireshark/pdml2html.xsl packets.xml > packets.html
root@kali:~# firefox packets.html &
[1] 3554
```

Чтобы открыть HTML-страницу в браузере, нужно обратить внимание на приведенное выше изображение и использовать следующую команду:

```
firefox packets.html &
```



?????? ??????? ????????????? ?????

Очень часто пользователь применяет Wireshark на определенном порту. А с помощью опции «**-f**» человек также может захватывать пакеты данных определенного порта в TShark. Это помогает лучше анализировать пакеты данных сети. Пользователи используют эту функцию для захвата TCP-порта номер 80:

```
tshark -i eth0 -c 5 -f "tcp port 80"
```

```
root@kali:~# tshark -i eth0 -c 5 -f "tcp port 80"
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
    1 0.000000000 192.168.0.137 → 216.58.196.99 TCP 66 44084 → 80 [ACK] Seq=1 Ack=1 Win=501 Le
n=0 TSval=1654711984 TSecr=2257786848
    2 0.000114735 192.168.0.137 → 216.58.196.99 TCP 66 44088 → 80 [ACK] Seq=1 Ack=1 Win=501 Le
n=0 TSval=1654711984 TSecr=1873411796
    3 0.000181040 192.168.0.137 → 216.58.196.99 TCP 66 44020 → 80 [ACK] Seq=1 Ack=1 Win=501 Le
n=0 TSval=1654711984 TSecr=1912130170
    4 0.082268726 216.58.196.99 → 192.168.0.137 TCP 66 [TCP ACKed unseen segment] 80 → 44084 [
ACK] Seq=1 Ack=2 Win=248 Len=0 TSval=2257797106 TSecr=1654660892
    5 0.082288921 216.58.196.99 → 192.168.0.137 TCP 66 [TCP ACKed unseen segment] 80 → 44020 [
ACK] Seq=1 Ack=2 Win=252 Len=0 TSval=1912140428 TSecr=1654660942
5 packets captured
```

???????? ??????

Экранный фильтр был представлен компанией Wireshark. Он помогает пользователям фильтровать захваченные пакеты данных или пакеты данных в реальном времени. С помощью этого фильтра человек может получить любой вид фильтра, который он хочет.

В данном сценарии пользователь применяет фильтр **GET request** для захвата только **GET request** из трафика. Используется команда:

```
tshark -i eth0 -c 5 -f "tcp port 80" -Y 'http.request.method == "GET" '
```

```
root@kali:~# tshark -i eth0  -f "tcp port 80" -Y 'http.request.method == "GET" '
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
   10 2.409241258 192.168.0.137 → 176.28.50.165 HTTP 445 GET / HTTP/1.1
   14 2.660232261 192.168.0.137 → 176.28.50.165 HTTP 465 GET /style.css HTTP/1.1
   18 2.916155632 192.168.0.137 → 176.28.50.165 HTTP 467 GET /images/logo.gif HTT
   22 4.140393060 192.168.0.137 → 176.28.50.165 HTTP 445 GET / HTTP/1.1
   26 4.381818253 192.168.0.137 → 176.28.50.165 HTTP 465 GET /style.css HTTP/1.1
   30 4.666101908 192.168.0.137 → 176.28.50.165 HTTP 467 GET /images/logo.gif HTT
   35 5.862037621 192.168.0.137 → 176.28.50.165 HTTP 445 GET / HTTP/1.1
   43 6.349299450 192.168.0.137 → 176.28.50.165 HTTP 465 GET /style.css HTTP/1.1
   49 6.659285080 192.168.0.137 → 176.28.50.165 HTTP 467 GET /images/logo.gif HTT
   56 8.452166611 192.168.0.137 → 176.28.50.165 HTTP 445 GET / HTTP/1.1
   61 8.747440415 192.168.0.137 → 176.28.50.165 HTTP 465 GET /style.css HTTP/1.1
   64 9.021046798 192.168.0.137 → 176.28.50.165 HTTP 467 GET /images/logo.gif HTT
```

?????????????? ??????

TShark собирает различные типы статистических данных и отображает их в результате после завершения чтения захваченного файла. Для этого будет использоваться параметр «**-z**» в TShark. Изначально, чтобы узнать обо всех различных опциях внутри параметра «**-z**», пользователь откроет TShark уже с включенной опцией, за которой будет следовать

ключевое слово **help**. Это даст человеку исчерпывающий список различных поддерживаемых форматов, как показано на рисунке ниже.

```
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
```

?????????? ???????? ??????????

Используя TShark, человек может создать иерархию на основе протоколов, в которой будет перечислено количество пакетов и байтов. Нужно использовать опцию «**io,phs**» в параметре «**-z**». В случае, когда фильтр не задан после опции «**io,phs**», статистика будет рассчитана для всех пакетов в радиусе видимости. Но если предусмотрен определенный фильтр, то TShark будет вычислять статистику для тех пакетов, которые соответствуют выбору, сделанному пользователем. Для демонстрации пользователь сначала захватил некоторый трафик и записал его содержимое в файл .pcap. Затем он будет использовать трафик из файла и сортировать эти данные прямо в иерархию протоколов. Читатели могут заметить,

что у пользователя есть определенное количество фреймов, размер пакетов в байтах и протокол, используемый для передачи.

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

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

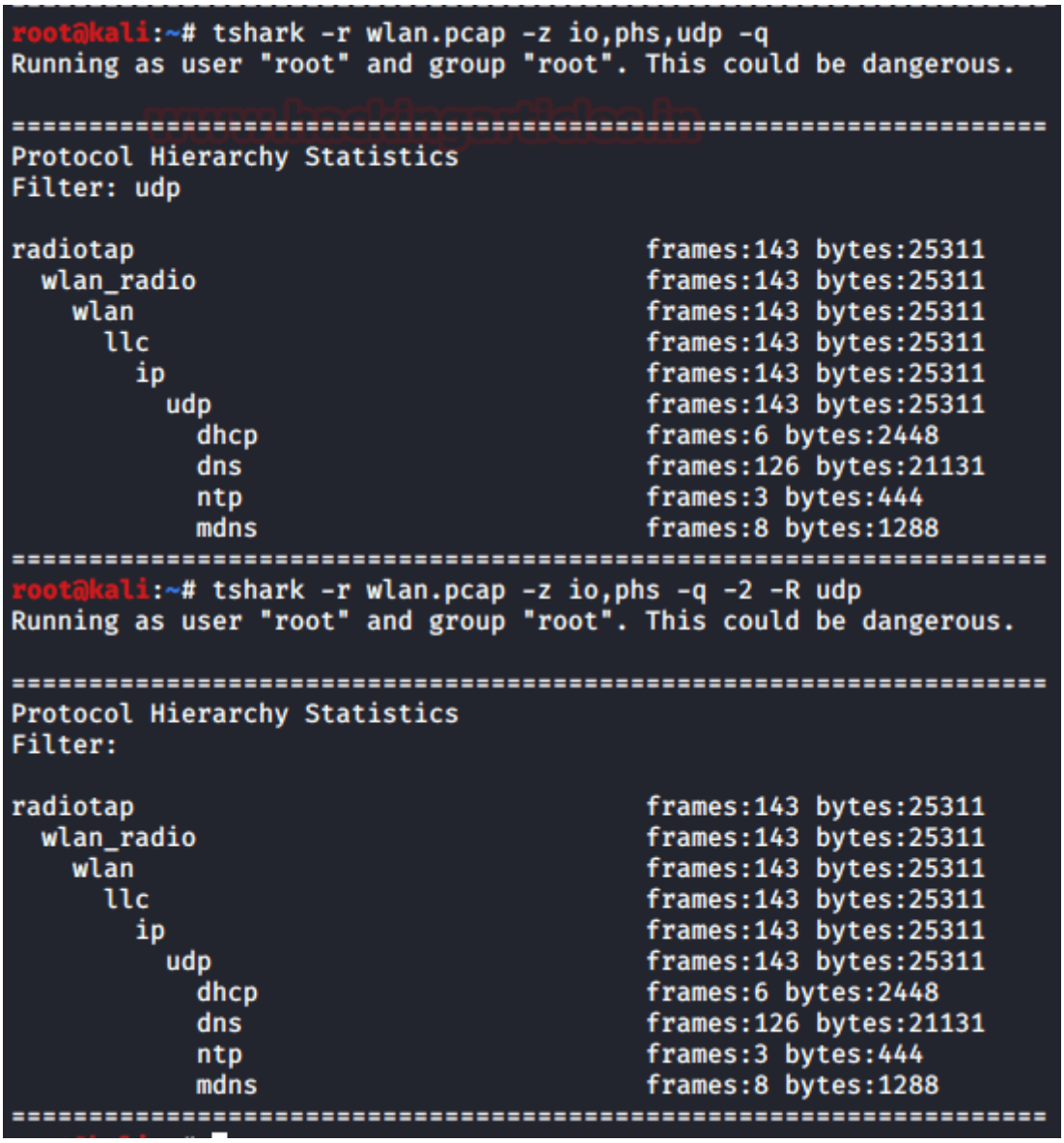
radiotap                                 frames:66690 bytes:15014549
  wlan_radio                             frames:66690 bytes:15014549
    wlan                                 frames:66690 bytes:15014549
      wlan                               frames:6873 bytes:1923747
      data                               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
===================================================================
```

?????? ??????? ??? ??????

Во время первого анализа пакета должен быть применен указанный фильтр (который использует синтаксис фильтров чтения/отображения, а не фильтров захвата). Пакеты, которые не соответствуют фильтру, не рассматриваются для будущих случаев. Этот параметр имеет смысл при нескольких проходах. Стоит обратить свое внимание, что поля, такие как «**response in frame #**»не могут быть использованы с этим фильтром, так как они не будут вычислены. Параметр «**-2**» выполняет двухпроходный анализ. Это приводит к тому, что TShark буферизует выходные данные до тех пор, пока не будет выполнен весь первый проход, но также заполняет поля, требующие будущих знаний, а также правильно вычисляет зависимости фреймов повторной сборки. Здесь пользователь может увидеть два

различных анализа, один из которых является анализом первого прохода, а второй – двухпроходным анализом.

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



?????? ???????? ?????

Следующий пункт, который помогает человеку разобраться со статистикой, — это «**Конечные точки**». Будет создана таблица в которой перечислены все конечные точки, которые можно было бы увидеть при захвате. Функция «**type**», которая может быть использована с параметром **endpoint**, укажет тип конечной точки, для которой человек хочет сгенерировать статистику.

В случае, если пользователь указал определенный фильтр, то расчеты статистики выполняются для этого конкретного фильтра. Таблица, подобная той, что сгенерирована на

изображении, показанном ниже, создается путем подбора однострочной формы каждой беседы и отображается напротив количества пакетов на байт в каждом направлении, а также общего количества пакетов на байт. Данные в этой таблице по умолчанию сортируются в соответствии с общим количеством фреймов.

```
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
```

?????? ?????

Стоит перейти к следующему разделу, который очень похож на предыдущий. Помогает со статистикой именно “conversation”. Будет создана таблица, в которой перечислены все разговоры, что можно было бы увидеть при захвате. Функция **type**, что используется с опцией **conversation**, укажет тип беседы, для которой человек хочет сгенерировать статистику.

Если пользователь установил определенный фильтр, то вычисления статистики выполняются для этого конкретного указанного фильтра. Таблица, сгенерированная путем подбора однострочной формы каждого разговора и отображаемая с количеством пакетов на байт в каждом направлении и общем количеством пакетов на байт, будет показана на экране. Данные сортируются в соответствии с общим количеством фреймов.

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

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

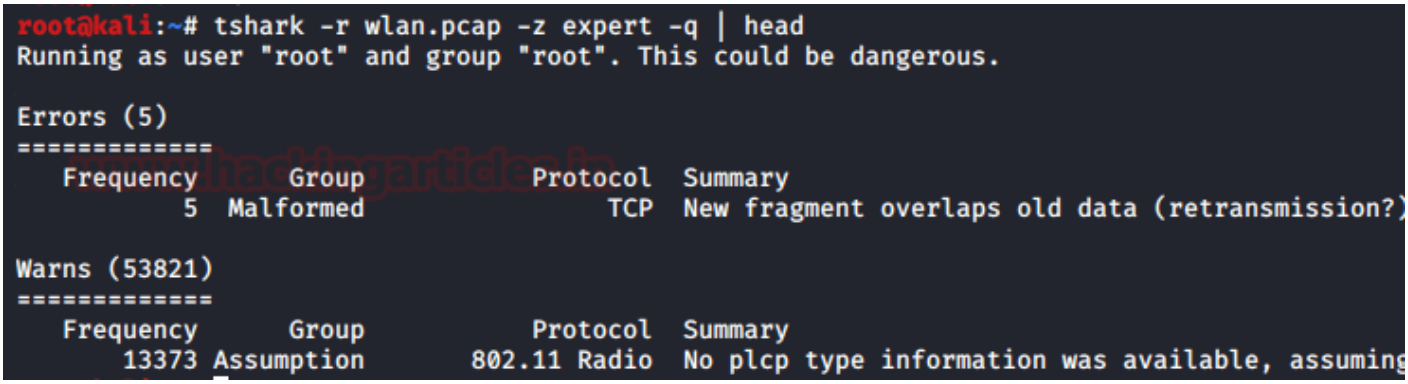
Errors (5)
==========
   Frequency      Group  Protocol  Summary
           5  Malformed       TCP  New fragment overlaps old data (retransmission?)

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

?????? ? ??????????? ??????

Модуль статистики TShark обладает ***экспертным режимом***. Он собирает огромное количество данных на основе полученной информации, а затем печатает ее в определенном порядке. Все эти данные сгруппированы в разделы, такие как ошибки, предупреждения и т.д. Человек также может использовать экспертный режим с определенным протоколом. В этом случае будут отображены все экспертные элементы конкретного протокола.

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



????????????? ???????

При этом варианте человек берет трафик из пакета, а затем прогоняет его через опцию «**http,tree**» с параметром «**-z**», чтобы подсчитать количество HTTP-запросов, их моды, а также код состояния. Это модульный подход, который очень легко понять и потом проанализировать свои действия. Здесь, в данном случае, пользователь взял пакет, который он захватил ранее, а затем прогнал его через опцию, что предоставила ему информацию о том, что в общей сложности было сгенерировано 126 запросов, из которых 14 вернули «**200 OK**». Это означает, что остальные из них либо выдали ошибку, либо были перенаправлены на другой сервер, выдав ответ серии 3XX.

```
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%
```

????? ???????

Поскольку идет речь о структуре похожей на дерево, стоит немного ее исследовать. У человека есть много способов, с помощью которых он может применить опцию «**tree**» в сочетании с другими функциями. Чтобы продемонстрировать это, человек решил использовать параметр длины пакета с параметром «tree». Это позволит отсортировать данные на основе размера пакетов, а затем сгенерировать таблицу с ними. Теперь эта таблица будет содержать не только длину пакетов, но и их количество. Минимальное значение длины будет находиться в диапазоне размеров пакетов. Также инструмент вычислит размер и процент пакетов внутри диапазона установленной длины.

```
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.

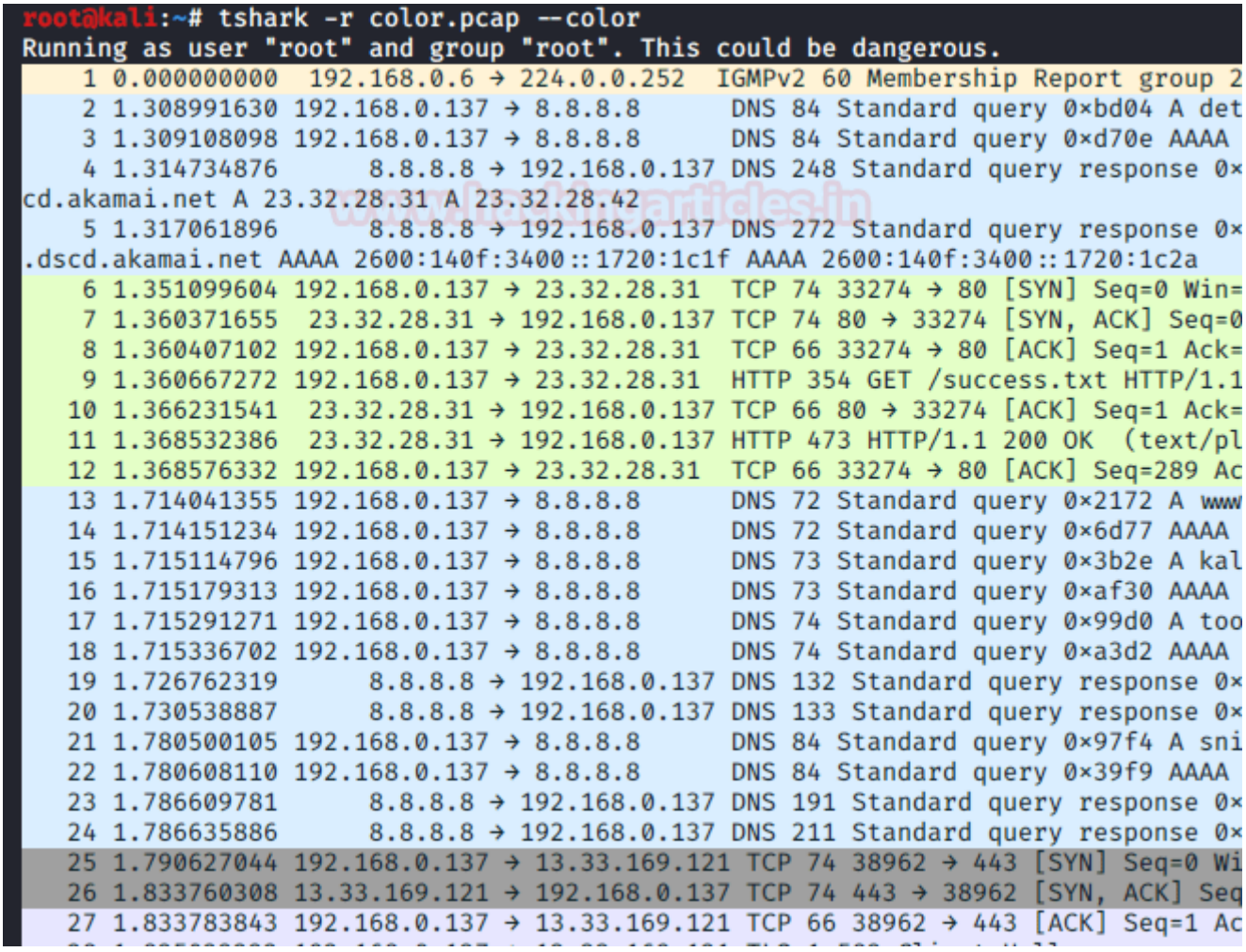
================================================================================
Packet Lengths:
Topic / Item       Count         Average       Min val       Max val       Rate (ms)     Percent
--------------------------------------------------------------------------------
Packet Lengths     66690         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%
```

?????? ???????? ?????? ?? ?????? ?????

Стоит обратить внимание: терминал должен поддерживать вывод цветов, чтобы эта опция работала правильно.

Человек может включить цвета для пакетов в соответствии со стандартными фильтрами Wireshark. В Windows цвета ограничены установленными вариантами. В данном случае пользователь может настроить цвета в соответствии с фильтром отображения. Это помогает быстро найти определенный пакет в группе подобных пакетов. Он также полезен при обнаружении рукопожатий в коммуникационном трафике. Его можно включить с помощью следующей команды:

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



?????? ???????????? ??????

По умолчанию TShark работает в режиме «***нескольких файлов***». Таким образом, TShark записывает данные в несколько файлов захвата. Когда первый файл захвата заполняется до определенной степени, инструмент переключается на следующий файл и так далее. Имена

файлов, которые человек хочет создать, можно указать с помощью параметра «**-w**». Количество файлов, данные о времени их появления будут объединены с именем, указанным рядом с параметром «-**w**», чтобы сформировать полное имя файла.

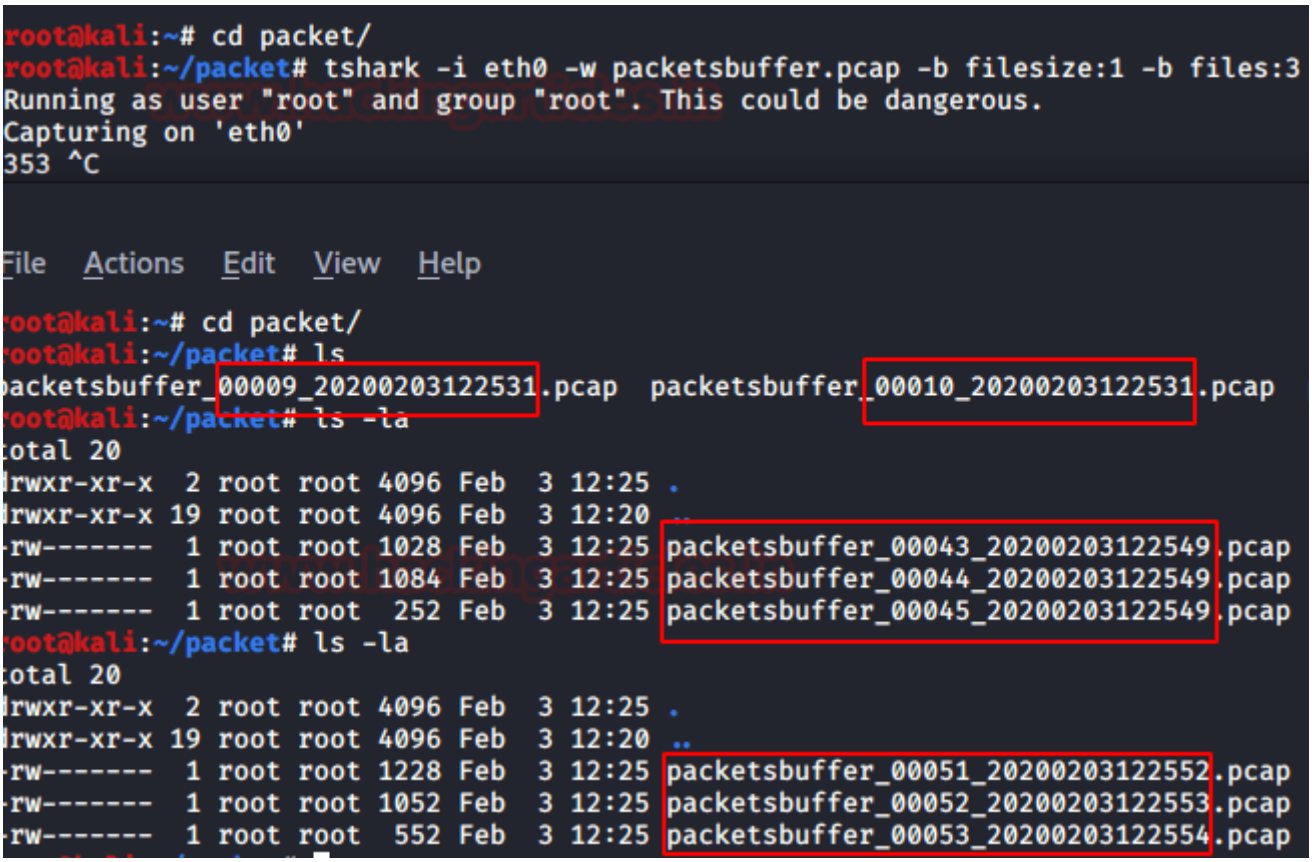
Опция «**files**» будет заполнять новые файлы до тех пор, пока не будет указано их количество. В этот момент TShark отбросит данные в первом файле и начнет запись в следующий файл. Если параметр «**files**» не включен, новые файлы заполняются до тех пор, пока одно из условий остановки захвата не будет выполнено или пока диск не будет до конца заполнен.

Существует множество критериев, по которым работает циклический буфер, но в данном случае человек использовал только 2 из них. Файлы и размер файла.

Файлы: значение начинается снова с первого файла после того, как было записано значение количества файлов (образуется циклический буфер). Это значение должно быть меньше 100000.

Размер файла: значение переключается на следующий файл после того, как первый достигнет установленного размера (КБ). Стоит обратить внимание, что максимальный размер файла ограничен ***2 гигабайтами***.

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



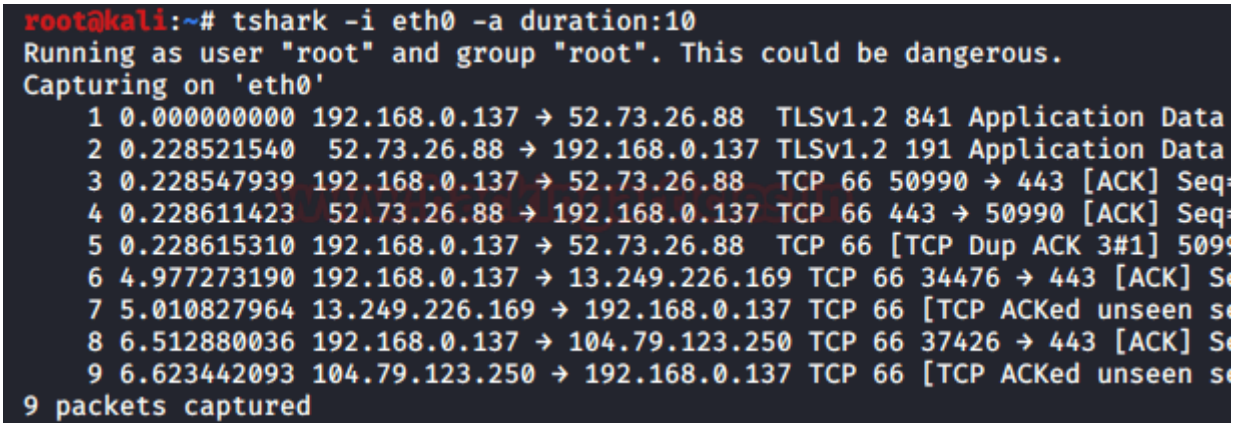
?????????????? ????????? ??????

Под огромным массивом опций есть одна, называемая **auto-stop**. Как следует из названия, с помощью нее можно автоматически остановить захват трафика после того, как установленные условия будут выполнены.

?????

Есть несколько вариантов, в данном случае человек использовал критерий продолжительности. Он уточнил продолжительность и ограничил ее 10-ю. Это значение выражается в секундах. Итак, после захвата пользователь видит, что за 10 секунд он захватил 9 пакетов.

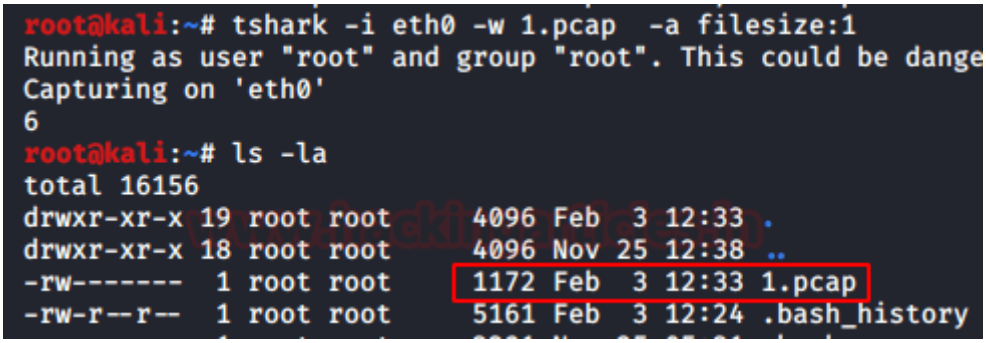
```
tshark -i eth0 -a duration:10
```



?????? ?????

Еще одним критерием для автоматической остановки работы является размер файла. TShark прекратит записи в указанный файл захвата после того, как он достигнет размера, установленного пользователем. В данном случае пользователь установил размер файла равным 1. Это значение считается в КБ. Он использовал команду для вывода списка каталогов, чтобы показать, что захват был прекращен, как только файл достиг размера 1 кб.

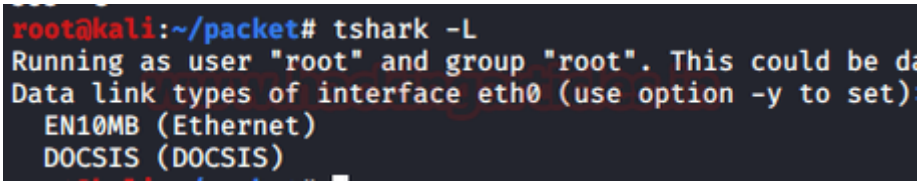
```
tshark -i eth0 -w 1.pcap -a filesize:1
```



???? ??????? ??? ???????? ??????

Пользователь также может изменить статистику захваченных данных трафика на основе типов каналов передачи данных. Для этого ему придется использовать независимый параметр «**-I**». В данном случае человек использовал его, чтобы показать, что у него есть каналы передачи данных, такие как **EN10MB**, указанные для трафика Ethernet.

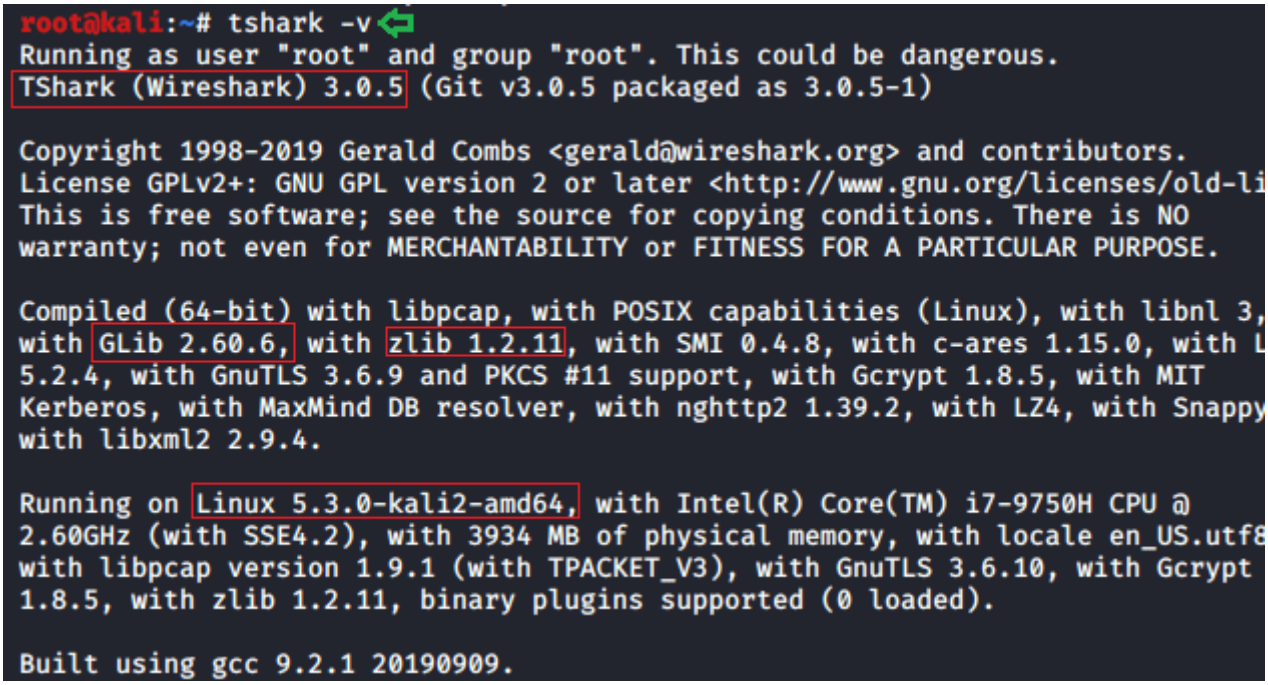
```
tshark -L
```



???????? ? ??????

Нужно ввести простую команду, чтобы читатели могли понять и соотнести все практические действия, выполненные в этой статье, с версией программы. Эта информация изображена на рисунке ниже. Данный параметр выводит информацию о версии установленного TShark.

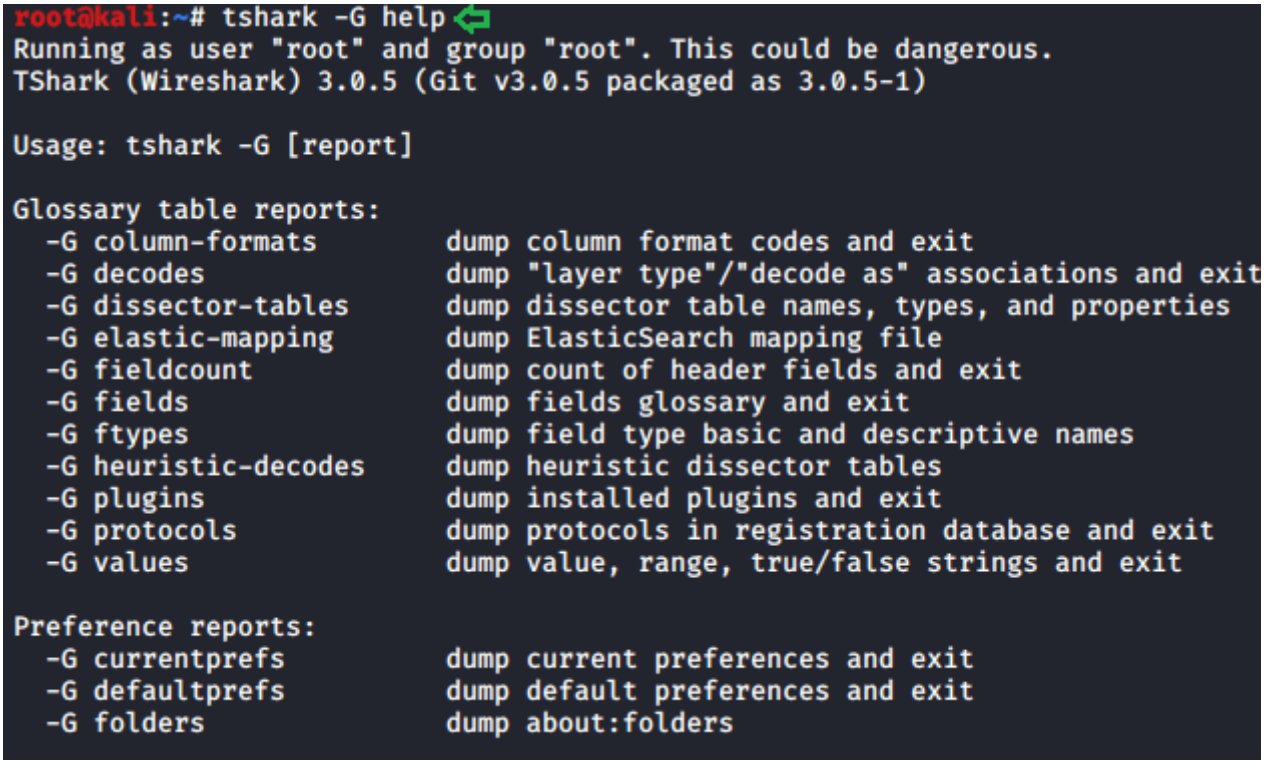
```
tshark -v
```



??????? ???????????

Во время любого захвата сети существует острая потребность в отчетности, чтобы человек мог поделиться результатами с командой, а также с начальством и иметь подтвержденные доказательства любой деятельности внутри нее. По тем же причинам TShark предоставил человеку прекрасный вариант (**-G**). С помощью этой опции TShark может распечатать список нескольких типов отчетов, которые будут сгенерированы. Официальное руководство по TShark использует слово ***Глоссарии*** для описания данных типов отчетности.

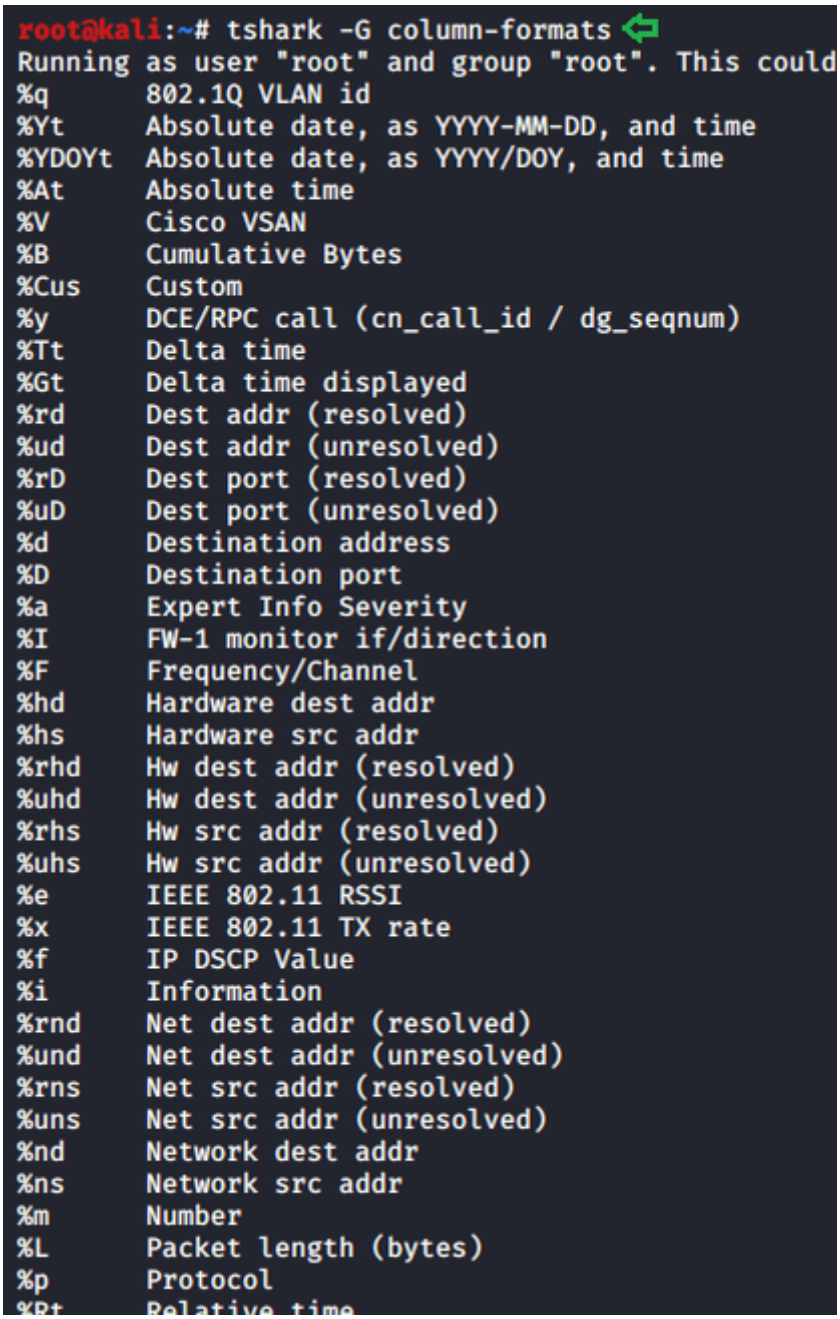
```
tshark -G help
```



??????? ????????

Конечно, есть опция форматирования столбцов, доступная в разделе отчетов TShark. Чтобы изучить его содержимое, человек выполнил команду, как показано на рисунке ниже. Читатели видят, что он печатает список с определенными знаками, которые могут быть использованы при создании отчета. Также есть идентификатор VLAN, дата, время, адрес назначения, порт назначения, длина пакета, протокол и другие данные.

```
tshark -G column-formats
```



?????????????

Эта опция способна сгенерировать 3 поля, связанные со слоями, а также декодированным протоколом. Существует ограничение, применяемое для одной записи в строке с этой опцией. Первое поле, имеющее значение «**s1ap.процесс.sout**» сообщает пользователю тип слоя сетевых пакетов. Далее он получает значение селектора в десятичном формате. Наконец, у человека есть расшифровка того, что было получено при захвате. Он использовал команду **head**, так как выходные данные были довольно большими, чтобы поместиться на одном скриншоте.

```
tshark -G decodes | head
```

```
root@kali:~# tshark -G decodes | head
Running as user "root" and group "root". This could be dangerous.
s1ap.proc.sout  17      s1ap
s1ap.proc.sout  3       s1ap
s1ap.proc.sout  6       s1ap
s1ap.proc.sout  23      s1ap
s1ap.proc.sout  9       s1ap
s1ap.proc.sout  48      s1ap
s1ap.proc.sout  43      s1ap
s1ap.proc.sout  29      s1ap
s1ap.proc.sout  4       s1ap
s1ap.proc.sout  21      s1ap
```

????????? ??????

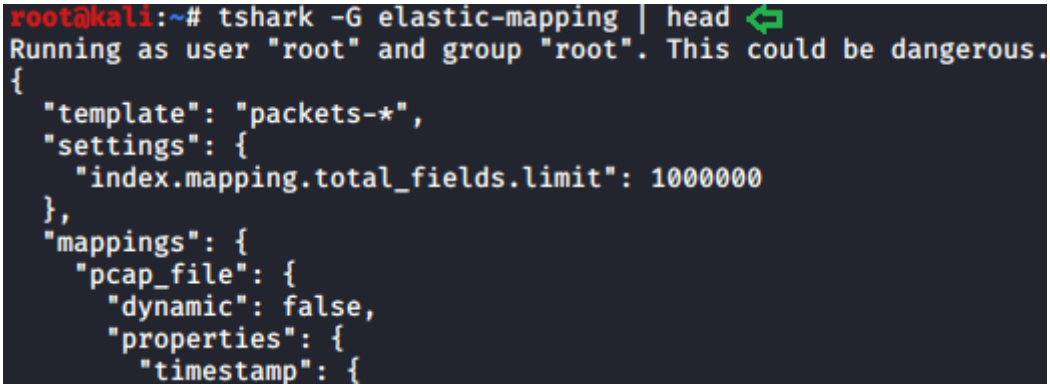
Большинство людей, читающих эту статью, уже знакомы с концепцией ***диссектора***. Если нет, то простыми словами Dissector — это синтаксический анализатор протокола. Выходные данные, генерируемые этой опцией, состоят из 6 полей. Начиная с имени таблицы диссектора, оно используется для таблицы диссектора в формате GUI. Далее, у человека уже есть тип и база для отображения имени протокола. Наконец, у него также есть возможность для декодирования полученной информации.

```
root@kali:~# tshark -G dissector-tables
Running as user "root" and group "root". This could be dangerous.
amqp.version    AMQP versions   FT_UINT8        BASE_DEC        AMQP    Decode As supported
ansi_637.tele_id        ANSI IS-637-A Teleservice ID    FT_UINT8        BASE_DEC        ANSI IS-637-A Te
ted
ansi_a.ota      IS-683-A (OTA)  FT_UINT8        BASE_DEC        ANSI BSMAP      Decode As not supported
ansi_a.pld      IS-801 (PLD)    FT_UINT8        BASE_DEC        ANSI BSMAP      Decode As not supported
ansi_a.sms      IS-637-A (SMS)  FT_UINT8        BASE_DEC        ANSI BSMAP      Decode As not supported
ansi_map.ota    IS-683-A (OTA)  FT_UINT8        BASE_DEC        ANSI MAP        Decode As not supported
ansi_map.pld    IS-801 (PLD)    FT_UINT8        BASE_DEC        ANSI MAP        Decode As not supported
ansi_map.tele_id        IS-637 Teleservice ID   FT_UINT8        BASE_DEC        ANSI MAP        Decode A
ansi_tcap.nat.opcode    ANSI TCAP National Opcodes      FT_UINT16       BASE_DEC        ANSI_TCAP
ansi_tcap.ssn   ANSI SSN        FT_UINT8        BASE_DEC        TCAP    Decode As not supported
arcnet.protocol_id      ARCNET Protocol ID      FT_UINT8        BASE_HEX        ARCNET  Decode As not su
aruba_erm.type  Aruba ERM Type  FT_NONE ARUBA_ERM       Decode As supported
atm.aal2.type   ATM AAL_2 type  FT_UINT32       BASE_DEC        ATM     Decode As supported
atm.aal5.type   ATM AAL_5 type  FT_UINT32       BASE_DEC        ATM     Decode As not supported
atm.cell_payload.vpi_vci        ATM Cell Payload VPI VCI        FT_UINT32       BASE_DEC        ATM
atm.reassembled.vpi_vci ATM Reassembled VPI VCI FT_UINT32       BASE_DEC        ATM     Decode As not su
awdl.tag.number AWDL Tags       FT_UINT8        BASE_DEC        AWDL    Decode As not supported
ax25.pid        AX.25 protocol ID       FT_UINT8        BASE_HEX        AX.25   Decode As not supported
bacapp.vendor_identifier        BACapp Vendor Identifier        FT_UINT8        BASE_HEX        BACapp
bacnet.vendor   BACnet Vendor Identifier        FT_UINT8        BASE_HEX        BACnet  Decode As not su
bacp.option     PPP BACP Options        FT_UINT8        BASE_DEC        PPP BACP        Decode As not su
bap.option      PPP BAP Options FT_UINT8        BASE_DEC        PPP BAP Decode As not supported
bcp_ncp.option  PPP BCP NCP Options     FT_UINT8        BASE_DEC        PPP BCP NCP     Decode As not su
bctp.tpi        BCTP Tunneled Protocol Indicator        FT_UINT32       BASE_DEC        BCTP    Decode A
```

Elastic mapping

Mapping – это очертание документов, хранящихся в индексе. **Elasticsearch** поддерживает различные типы данных для полей в документе. Параметр **elastic-mapping** в TShark выводит данные, хранящиеся внутри файла сопоставления ElasticSearch. Из-за большого количества данных для печати пользователь решил также использовать команду **head**.

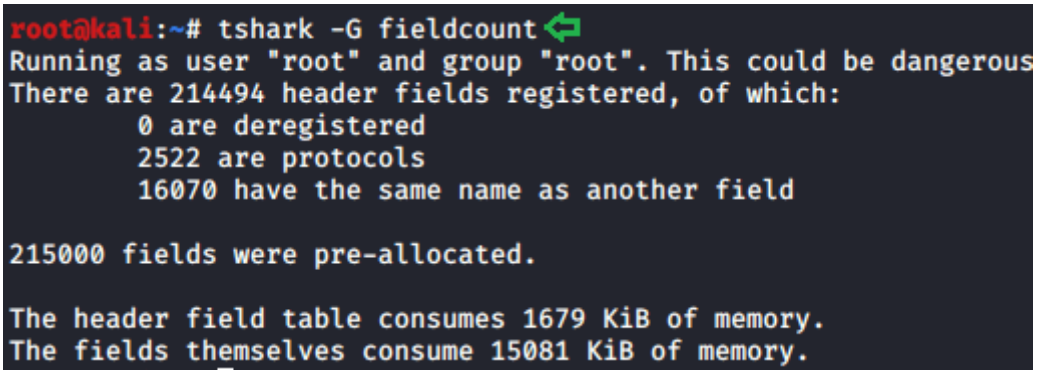
```
tshark -G elastic-mapping | head
```



?????????? ?????

В сетевой трассировке бывают моменты, когда нужно получить количество полей заголовка, перемещающихся в любой момент. В таких случаях TShark может помочь. С помощью опции **fieldcount** человек способен легко распечатать количество полей заголовка. Как можно увидеть на изображении, приведенном ниже, у пользователя есть 2522 протокола и 215000 полей, которые были предварительно распределены.

```
tshark -G fieldcount
```

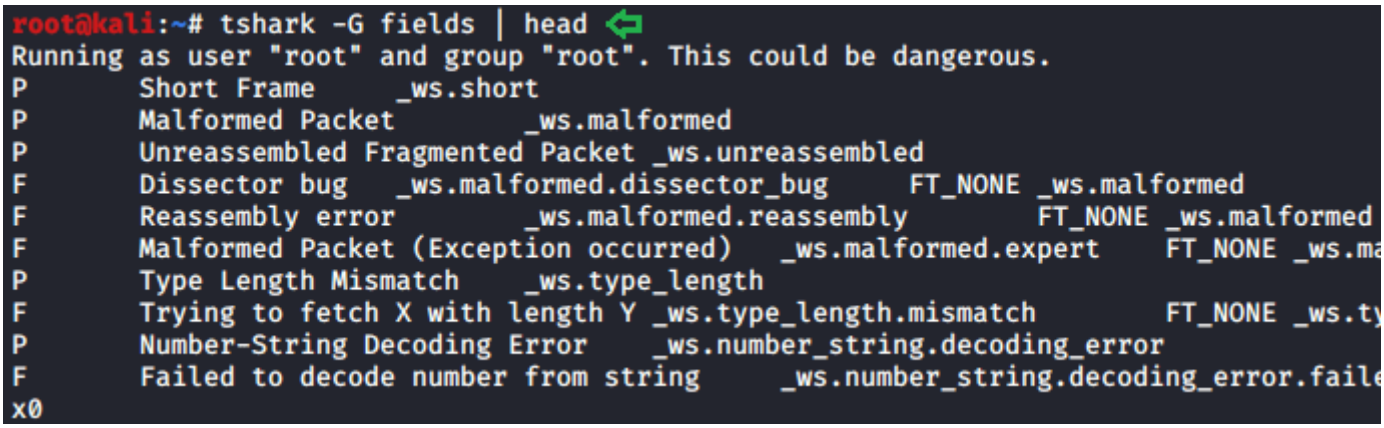


????

TShark также может получить содержимое регистрационной базы данных. Выходные данные, генерируемые этой опцией, не так легко интерпретировать, как другие. Некоторые пользователи могут использовать любой другой инструмент синтаксического анализа для получения более точного результата. Каждая запись в выходных данных является протоколом или файлом заголовка. Это можно различить по первому полю записи. Если поле P, то это протокол, а если F, то это поле заголовка. В случае с протоколами есть еще 2 поля. Один из них говорит о протоколе, а другие поля показывают аббревиатуру, используемую для указанного протокола. В случае с заголовком ситуация немного отличается. Более того, есть еще 7 полей. Пользователь получает имя, аббревиатуру, тип, аббревиатуру

родительского протокола, базу для отображения файлов, поле для описания рекламы, битовую маску.

```
tshark -G fields | head
```



???????? ????

TShark также помогает человеку создавать отчеты, концентрирующиеся вокруг основных типов сетевых протоколов. Это сокращенно называется **ftype**. Этот тип отчета состоит всего из 2 полей. Один для FTYPE, а другой для его описания.

```
tshark -G ftypes
```

```
root@kali:~# tshark -G ftypes
Running as user "root" and group "root". This could be dangerous
FT_NONE Label
FT_PROTOCOL     Protocol
FT_BOOLEAN      Boolean
FT_CHAR Character, 1 byte
FT_UINT8        Unsigned integer, 1 byte
FT_UINT16       Unsigned integer, 2 bytes
FT_UINT24       Unsigned integer, 3 bytes
FT_UINT32       Unsigned integer, 4 bytes
FT_UINT40       Unsigned integer, 5 bytes
FT_UINT48       Unsigned integer, 6 bytes
FT_UINT56       Unsigned integer, 7 bytes
FT_UINT64       Unsigned integer, 8 bytes
FT_INT8 Signed integer, 1 byte
FT_INT16        Signed integer, 2 bytes
FT_INT24        Signed integer, 3 bytes
FT_INT32        Signed integer, 4 bytes
FT_INT40        Signed integer, 5 bytes
FT_INT48        Signed integer, 6 bytes
FT_INT56        Signed integer, 7 bytes
FT_INT64        Signed integer, 8 bytes
FT_IEEE_11073_SFLOAT    IEEE-11073 Floating point (16-bit)
FT_IEEE_11073_FLOAT     IEEE-11073 Floating point (32-bit)
FT_FLOAT        Floating point (single-precision)
FT_DOUBLE       Floating point (double-precision)
FT_ABSOLUTE_TIME        Date and time
FT_RELATIVE_TIME        Time offset
FT_STRING       Character string
FT_STRINGZ      Character string
FT_UINT_STRING  Character string
FT_ETHER        Ethernet or other MAC address
FT_BYTES        Sequence of bytes
FT_UINT_BYTES   Sequence of bytes
FT_IPv4 IPv4 address
FT_IPv6 IPv6 address
FT_IPXNET       IPX network number
FT_FRAMENUM     Frame number
FT_PCRE Compiled Perl-Compatible Regular Expression (GRegex) obj
FT_GUID Globally Unique Identifier
```

?????????????? ??????????????

Сортировка диссекторов на основе эвристического декодирования – это одна из тех вещей, которые должны быть легко доступны. По той же причине у человека есть возможность эвристического декодирования в TShark. Эта опция дает возможность для печати всех эвристических декодов, которые в данный момент установлены. Состоит из 3-х полей.

```
tshark -G heuristic-decodes
```



???????

Плагины — это очень важный тип опций, который был интегрирован с опциями отчетности Tshark. Как следует из названия, он печатает имя всех установленных плагинов. Поле, из которого делается этот отчет, состоит из библиотеки, версии, типа плагинов и пути, по которому они находятся.

```
tshark –G plugins
```

```
root@kali:~# tshark -G plugins
Running as user "root" and group "root". This could be dangerous.
ethercat.so             0.1.0   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
gryphon.so              0.0.4   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
irda.so                 0.0.6   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
mate.so                 1.0.1   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
opcua.so                1.0.0   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
profinet.so             0.2.4   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
stats_tree.so           0.0.1   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
transum.so              2.0.4   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
unistim.so              0.0.2   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
usbdump.so              0.0.1   file type       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
wimax.so                1.2.0   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
wimaxasncp.so           0.0.1   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
wimaxmacphy.so          0.0.1   dissector       /usr/lib/x86_64-linux-gnu/wireshark/plugins/3
```

?????????

Если читатели хотят знать подробности о протоколах, которые записываются в базу данных, они могут использовать параметр **protocols**. Этот вывод также немного не читабелен, так что пользователь может использовать любой сторонний инструмент, чтобы составить отчет. Этот параметр выводит данные в 3-х полях. У человека есть имя протокола, краткое имя и имя фильтра.

```
tshark –G protocols | head
```

```
root@kali:~# tshark -G protocols | head
Running as user "root" and group "root". This could be dangerous.
Lua Dissection  Lua Dissection  _ws.lua
Expert Info     Expert  _ws.expert
IEC 60870-5-104-Apci    104apci 104apci
IEC 60870-5-104-Asdu    104asdu 104asdu
29West Protocol 29West  29west
Pro-MPEG Code of Practice #3 release 2 FEC Protocol     2dparityfec     2dparityfec
3Com XNS Encapsulation  3COMXNS 3comxns
3GPP2 A11       3GPP2 A11       a11
IPv6 over Low power Wireless Personal Area Networks     6LoWPAN 6lowpan
802.11 radio information        802.11 Radio    wlan_radio
```

????????

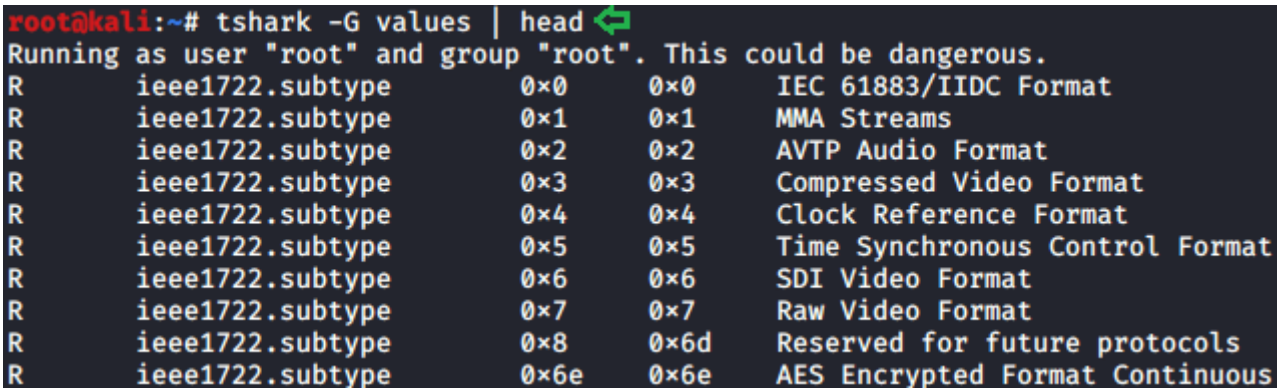
Отчет состоит из строк значений, строк диапазона, строк true/false. Здесь доступны три типа записей. Первое поле может состоять из одного из этих трех символов:

- V: строки значений;
- R: строки диапазона;
- T: true/false строки.

Кроме того, в строках значений есть аббревиатура поля, значение и сама строка. В строках диапазона есть те же значения, за исключением того, что они содержат информацию,

связанную с нижней и верхней границей строки.

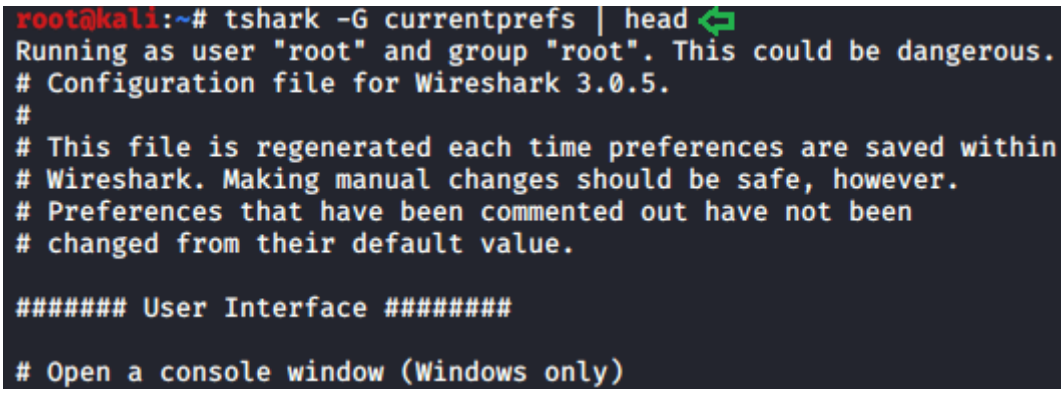
```
tshark –G values | head
```



??????????

В случае, если пользователю требуется пересмотреть текущие настройки, установленные в системе, он может использовать параметры **currentprefs** для выбора предпочтений, сохраненных в файле.

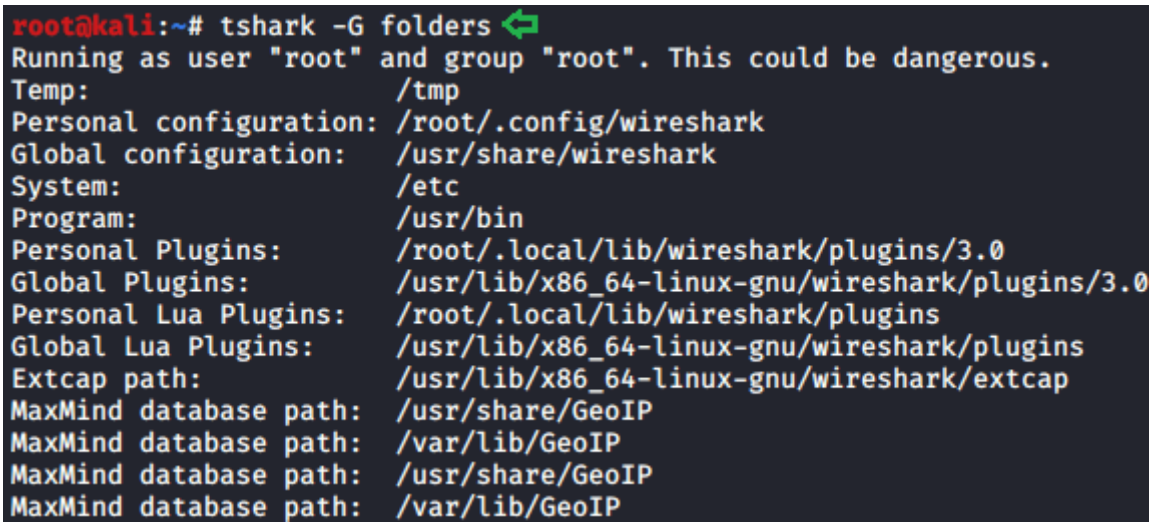
```
tshark –G currentprefs | head
```



?????

Предполагается, что пользователь хочет вручную изменить конфигурацию или получить информацию о программе. Ему нужен путь к этим файлам, чтобы взглянуть на них. Здесь пригодится опция **folders**.

```
tshark –G folders
```



Поскольку читатели так много знают о TShark, будет несправедливо, если пользователь не расскажет об инструменте, который сильно зависит от данных TShark. Пора поговорить о **PyShark**.

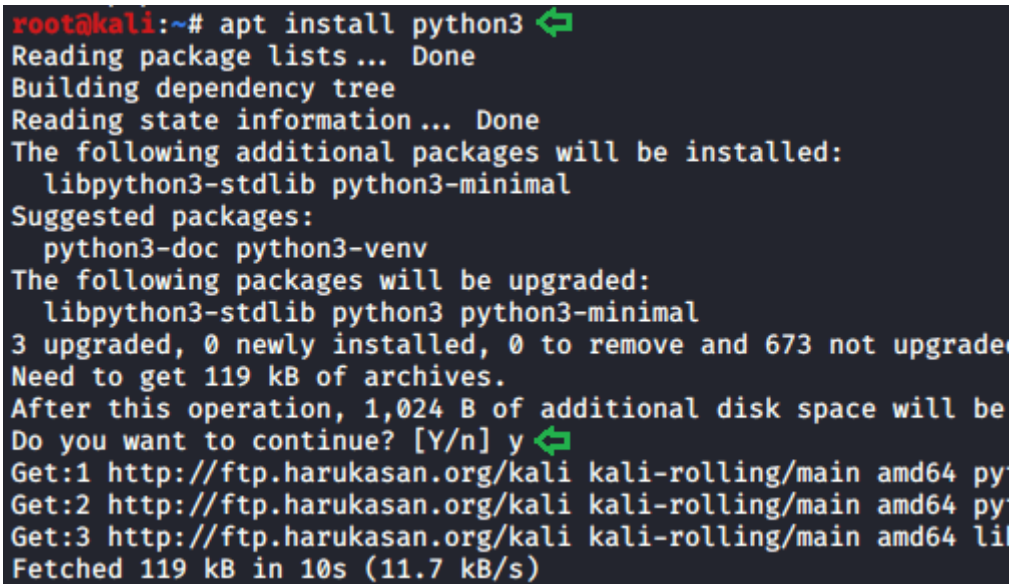
PyShark

По сути, это Shell, основанный на Python. Его функциональность заключается в том, что он позволяет анализировать пакеты Python с помощью диссекторов TShark. Многие инструменты выполняют одну и ту же работу более или менее хорошо, но разница заключается в том, что этот инструмент может экспортировать XML для использования его синтаксического анализа.

?????????

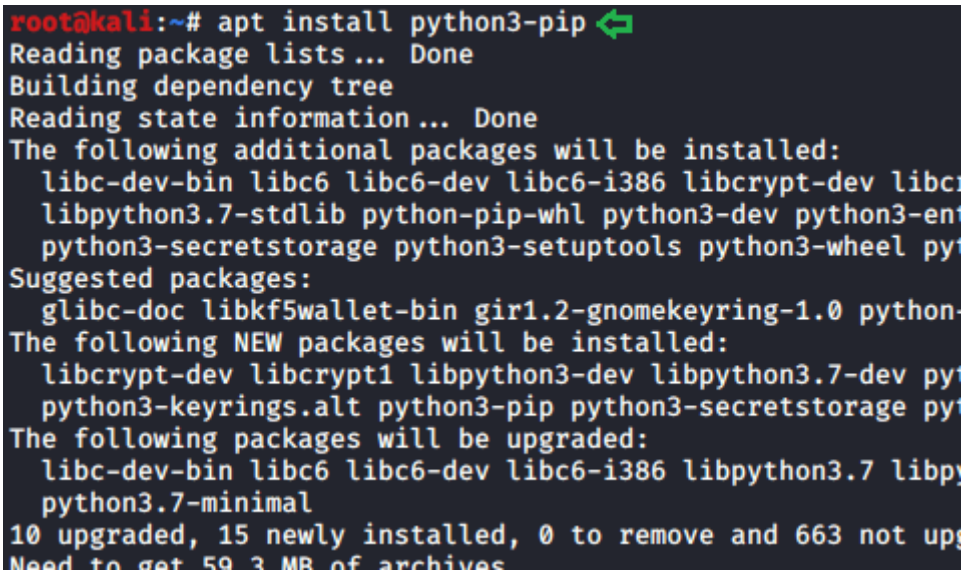
Поскольку PyShark был разработан с использованием Python 3, нужно установить его, как показано на рисунке ниже.

```
apt install python3
```



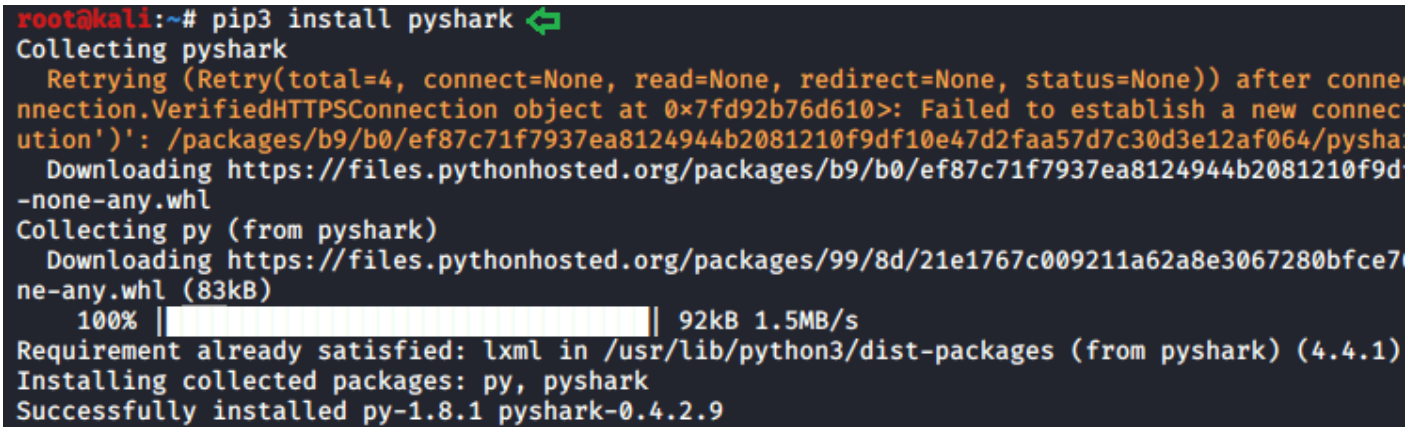
PyShark доступен через pip. Но у пользователя нет pip для Python 3, поэтому также нужно установить и его.

```
apt install python3-pip
```



Поскольку у человека уже есть Python3 с pip, он установит Pyshark с помощью команды. Он также может установить PyShark, клонировав git и запустив программу установки.

```
pip3 install pyshark
```



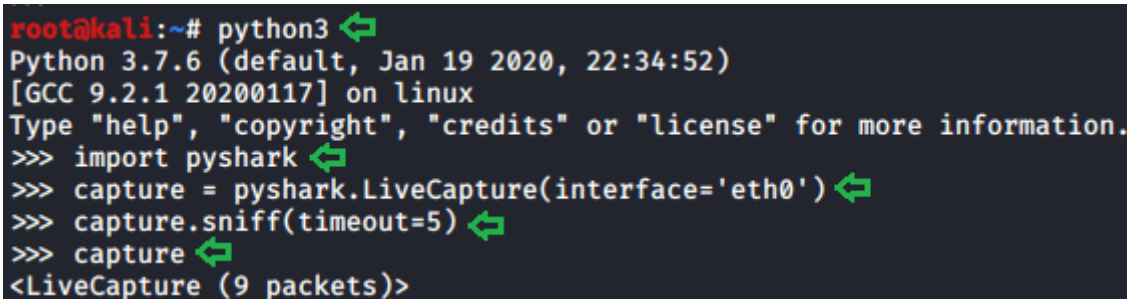
??????

Теперь, чтобы начать, нужен интерпретатор Python. Чтобы получить его, пользователь пишет **python3** и нажимает **Enter**. Теперь, когда у него есть интерпретатор, самое первое, что нужно сделать, – это импортировать PyShark. Затем он определяет сетевой интерфейс для захвата. Также человек определит значение параметра **timeout**.

Наконец, пользователь начинает захват. Здесь он видит, что были захвачены 9 пакетов.

```
python3
```

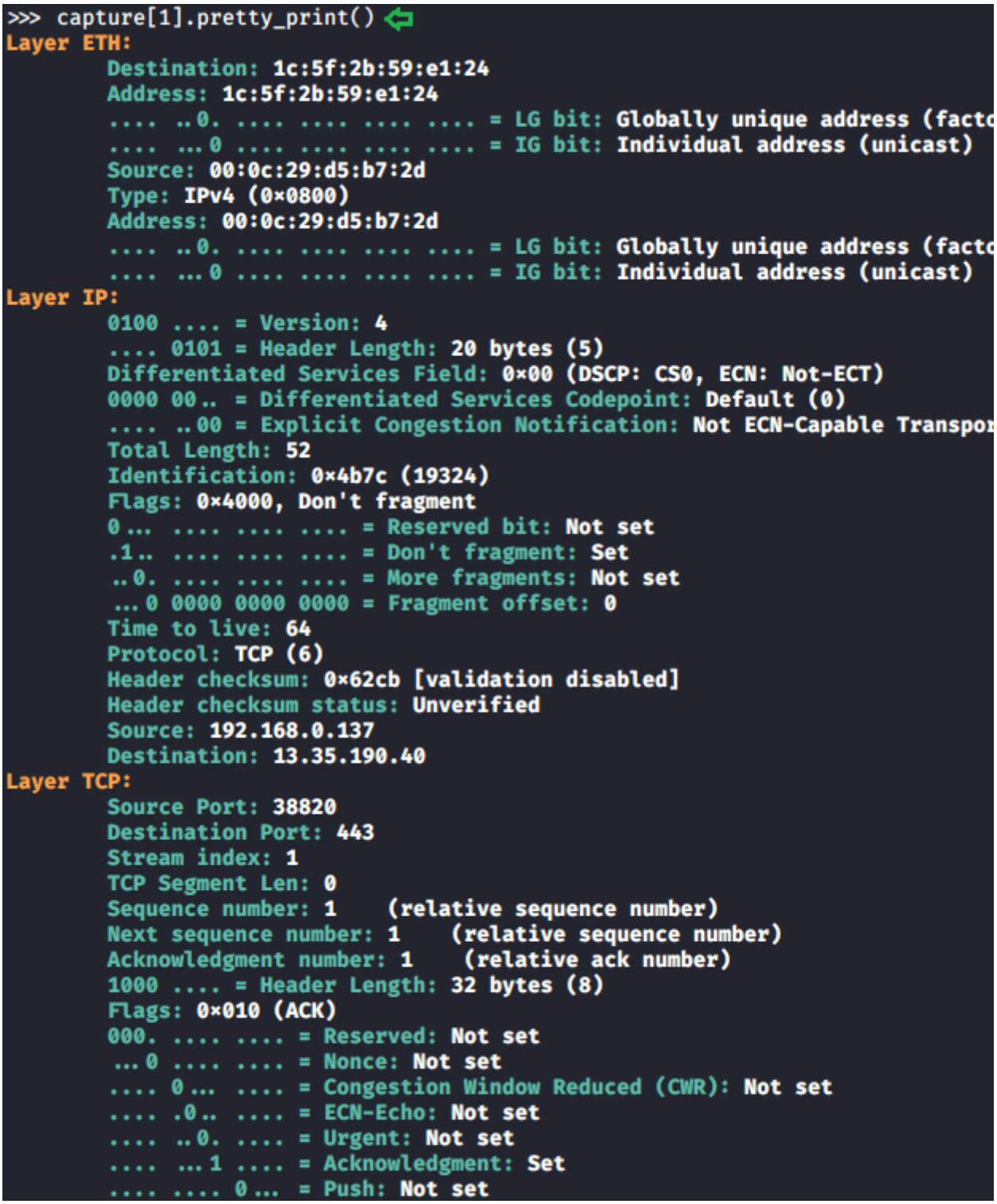
```
import pyshark
capture = pyshark.LiveCapture(interface='eth0')
capture.sniff(timeout=5)
capture
```



???????? ???????????

Существует несколько способов, которыми PyShark может представлять данные внутри захваченного пакета. В предыдущей практике были захвачены 9 пакетов. Стоит взглянуть на первый пакет, который был захвачен с помощью PyShark. Здесь пользователь может увидеть, что у него есть послойный анализ с уровнем ETH, уровнем IP и уровнем TCP.

```
capture[1].pretty_print()
```

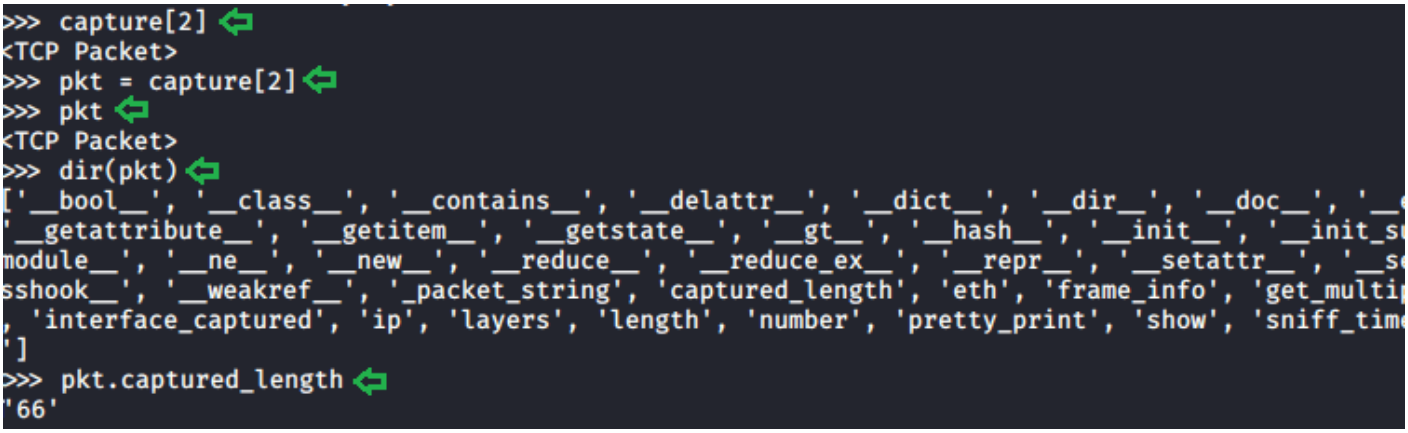


?????????? ???? ?????

При захвате пользователь увидел некоторые данные, которые состоят из нескольких атрибутов. Эти атрибуты нуждаются в полях для хранения. Чтобы исследовать данную область, человек будет использовать функцию **dir** в Python. Человек взял пакет, а затем определил переменную с именем **pkt** со значением этого пакета и сохранил все данные. Затем, используя функцию dir, он увидел, что исследовал поля внутри этого конкретного случая. Он понимает, что есть функция **pretty_print**. Также есть одно поле с именем **captured_length** для чтения, в которое он запишет имя переменной, за которым следует

имя поля с точкой (.) между ними, как показано на рисунке ниже.

```
capture[2]
pkt = capture[2]
pkt
dir(pkt)
pkt.captured_length
```



????, ???? Src ? Dst

Когда пользователь перечислил поля, он увидел, что у него есть еще одно поле с именем **layers**. Человек прочитает его содержимое, чтобы узнать, что у него есть 3 слоя в этом захвате. Теперь, чтобы заглянуть в определенный слой, нужно получить его поля. Для этого пользователь снова будет использовать функцию **dir**. Человек уже использовал функцию dir на слое ETH, как показано на рисунке ниже. У него есть поле с именем **src**, что означает источник, и **dst**, что означает пункт назначения. Он проверил значение в этих полях, чтобы найти физический адрес источника и пункт назначения.

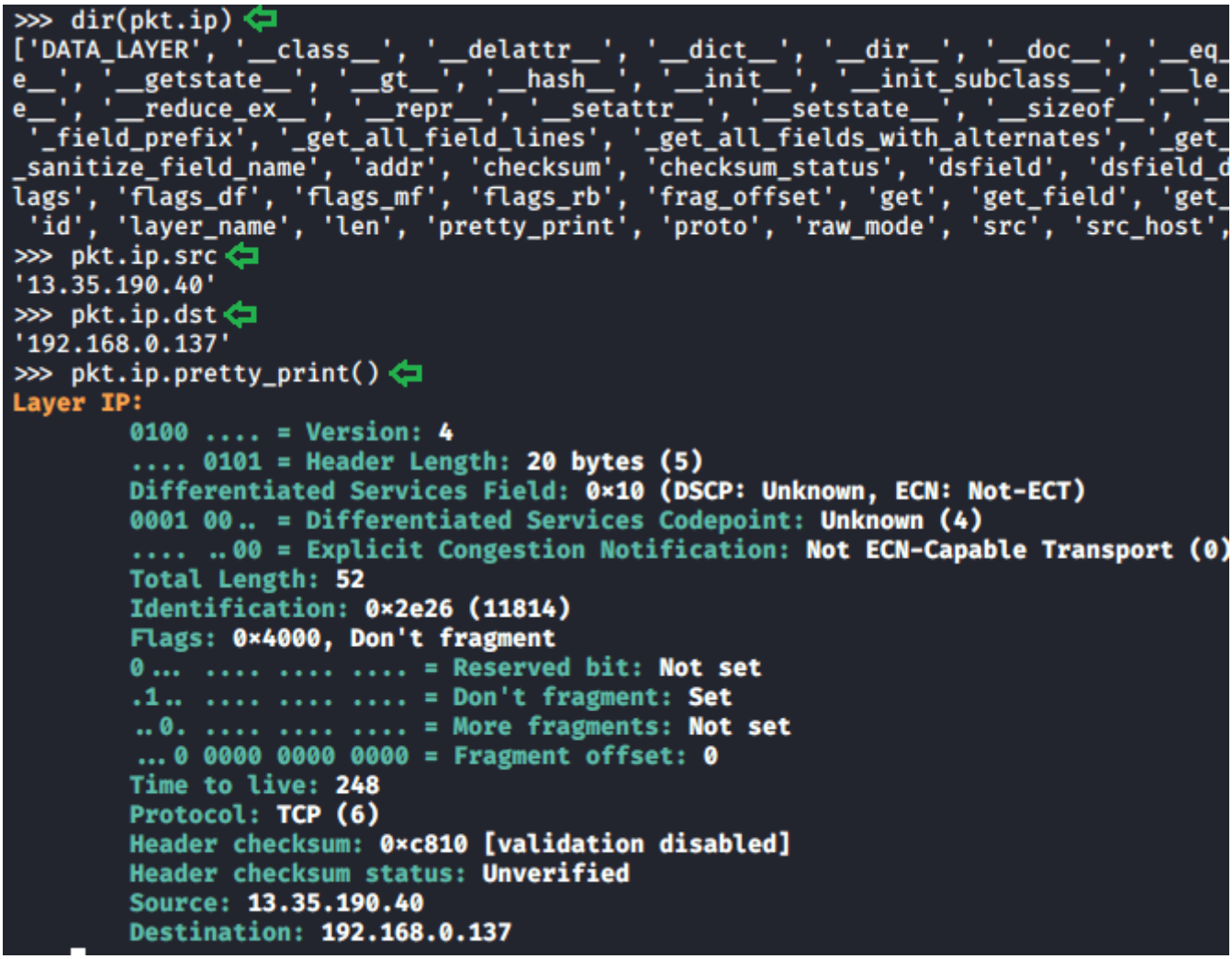
```
pkt.layers
pkt.eth.src
pkt.eth.dst
pkt.eth.type
```



Для следующего шага нужны поля IP-пакета. Человек использовал функцию **dir** на IP-слое, а затем применил поля **src** и **dst**.

Он видит, что есть IP-адрес, так как это IP-уровень. Поскольку уровень Ethernet работает на MAC-адресах, он хранит MAC-адреса источника и назначения, которые изменяются, когда человек переходит к IP-уровню.

```
dir(pkt.ip)
pkt.ip.src
pkt.ip.dst
pkt.ip.pretty_print()
```

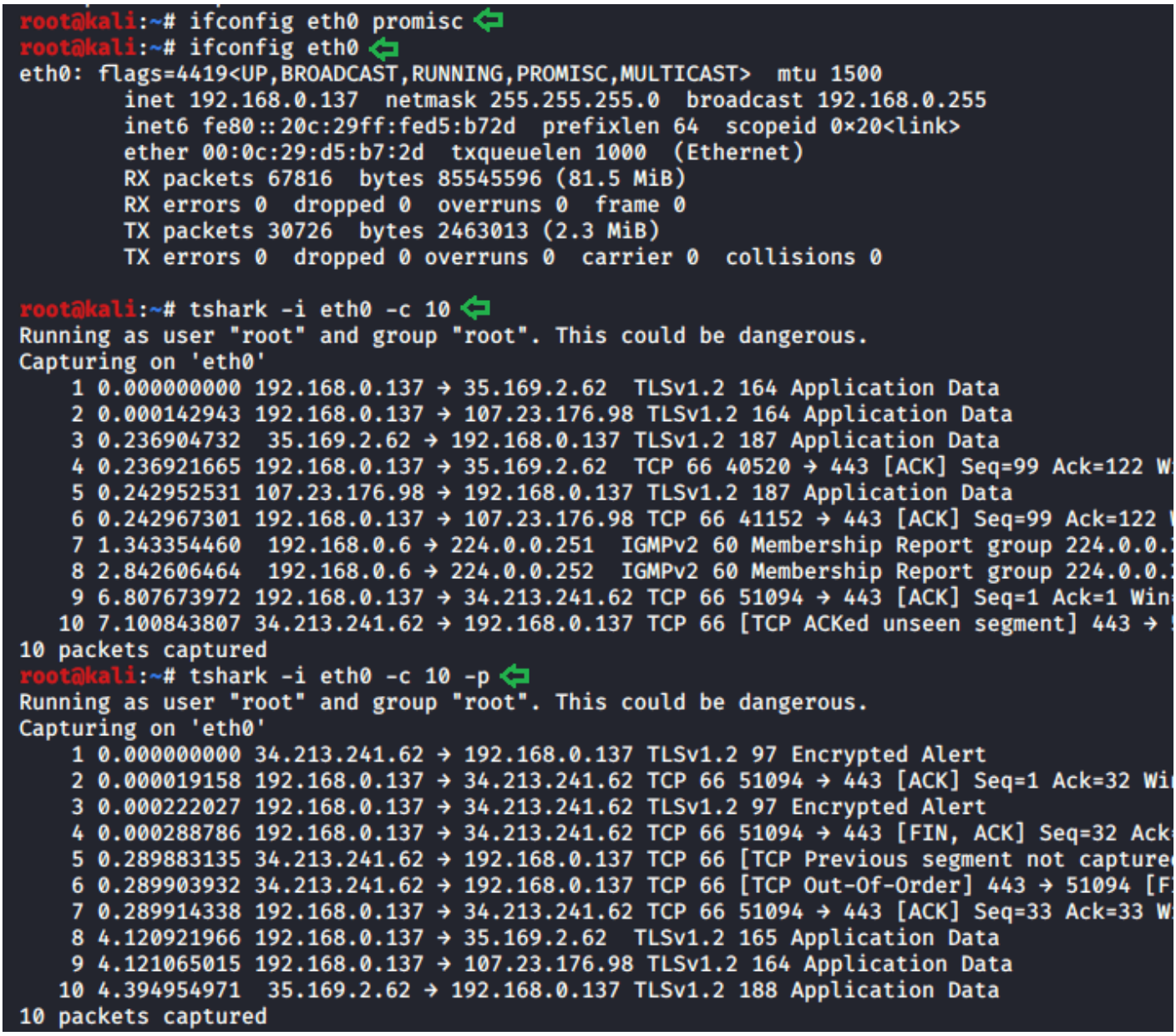


Аналогично, человек может использовать функцию **dir** и значение поля на любом слое захвата. Это значительно облегчает проведение захвата.

?????? **Promisc**

В предыдущих разделах читатели узнали о режиме promisc, который означает, что сетевая карта будет передавать все полученные фреймы в операционную систему для обработки, в отличие от традиционного режима работы, при котором только фреймы, предназначенные для MAC-адреса сетевой карты или широковещательного адреса, будут передаваться на компьютер. Как правило, этот режим используется для снифинга всего трафика. Но появилась загвоздка, когда пользователь настроил сетевую интерфейсную карту для работы в режиме **promisc**. Таким образом, при захвате трафика на TShark пользователь может переключаться между обычным захватом и захватом promisc с помощью параметра «**-p**», как показано на рисунке ниже.

```
ifconfig eth0 promisc
ifconfig eth0
tshark -i eth0 -c 10
tshark -i eth0 -c 10 -p
```



Важно! Информация исключительно в учебных целях. Пожалуйста, соблюдайте законодательство и не применяйте данную информацию в незаконных целях.

?????? ???????????? ??????????

- Руководство по TShark