

# Сетевые инструменты

- [Руководство по TShark](#)

# Руководство по 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.

## Сетевой трафик

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.

## Знакомство с 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

```

## Список интерфейсов

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
```

```
root@kali:~# tshark -D
Running as user "root" and group "root". This could be dangerous.
1. eth0
2. lo (Loopback)
3. any
4. nflog
5. nfqueue
6. ciscodump (Cisco remote capture)
7. dpauXmon (DisplayPort AUX channel monitor capture)
8. randpkt (Random packet generator)
9. sdjournal (systemd Journal Export)
10. sshdump (SSH remote capture)
11. udpdump (UDP Listener remote capture)
```

## Захват трафика

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
```

```
root@kali: ~  
File Actions Edit View Help  
root@kali:~# 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=117 ms  
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=2 ttl=54 time=181 ms  
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=3 ttl=54 time=249 ms  
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=4 ttl=54 time=131 ms  
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=5 ttl=54 time=210 ms  
64 bytes from 104.28.7.89 (104.28.7.89): icmp_seq=6 ttl=54 time=161 ms  
^C  
--- www.hackingarticles.in ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5008ms  
rtt min/avg/max/mdev = 117.326/174.819/249.328/45.171 ms  
root@kali:~#  
  
root@kali: ~  
File Actions Edit View Help  
root@kali:~# tshark -i eth0  
Running as user "root" and group "root". This could be dangerous.  
Capturing on 'eth0'  
1 0.000000000 192.168.0.137 → 8.8.8.8 DNS 82 Standard query 0x9c1a A www.hackingarticles.in  
2 0.000157409 192.168.0.137 → 8.8.8.8 DNS 82 Standard query 0x9620 AAAA www.hackingarticles.in  
3 0.113177929 8.8.8.8 → 192.168.0.137 DNS 114 Standard query response 0x9c1a A www.hackingarticles.  
A 104.28.7.89 A 104.28.6.89  
4 0.113200970 8.8.8.8 → 192.168.0.137 DNS 138 Standard query response 0x9620 AAAA www.hackingarticl  
.in AAAA 2606:4700:3031::681c:759 AAAA 2606:4700:3033::681c:659  
5 0.113563758 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=1/256, ttl=64  
6 0.230877040 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=1/256, ttl=54 (re  
est in 5)  
7 0.231050335 192.168.0.137 → 8.8.8.8 DNS 84 Standard query 0xf48f PTR 89.7.28.104.in-addr.arpa  
8 0.290869104 8.8.8.8 → 192.168.0.137 DNS 179 Standard query response 0xf48f No such name PTR 89.7.  
.104.in-addr.arpa SOA cruz.ns.cloudflare.com  
9 1.115479483 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=2/512, ttl=64  
10 1.296199640 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=2/512, ttl=54 (re  
est in 9)  
11 2.117862984 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=3/768, ttl=64  
12 2.367168921 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=3/768, ttl=54 (re  
est in 11)  
13 3.118443326 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=4/1024, ttl=64  
14 3.249467028 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=4/1024, ttl=54 (r  
uest in 13)  
15 4.120154691 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=5/1280, ttl=64  
16 4.330051501 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=5/1280, ttl=54 (r  
uest in 15)  
17 5.041217255 Vmware_d5:b7:2d → D-LinkIn_59:e1:24 ARP 42 Who has 192.168.0.1? Tell 192.168.0.137  
18 5.121457089 192.168.0.137 → 104.28.7.89 ICMP 98 Echo (ping) request id=0x0aaf, seq=6/1536, ttl=64  
19 5.142766220 D-LinkIn_59:e1:24 → Vmware_d5:b7:2d ARP 60 192.168.0.1 is at 1c:5f:2b:59:e1:24  
20 5.281995384 104.28.7.89 → 192.168.0.137 ICMP 98 Echo (ping) reply id=0x0aaf, seq=6/1536, ttl=54 (r  
uest in 18)
```

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



сниффинг пакетов. Такое имеет 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=0x1b86, seq=95/2432
  2 0.136847861 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=95/2432
  3 0.767140702 fe80::8dcf:3961:7c07:7841 → ff02::fb MDNS 180 Standard query 0x0000 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 0x0000 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=0x1b86, seq=96/2457
  7 1.232830355 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=96/2457
  8 2.002672796 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=97/2483
  9 2.534998232 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=97/2483
 10 3.003729111 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=98/2508
 11 3.151781403 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=98/2508
 12 4.005684733 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=99/2534
 13 4.221686910 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, 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=0x1b86, seq=100/2
16 5.132027301 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=100/256

```

## Количество пакетов

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 «-с», 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
```

## Чтение и редактирование файла

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
```

```
root@kali:~# tshark -i eth0 -c 10 -w packets.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
10
root@kali:~# tshark -r packets.pcap
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=295/9985, ttl=64
 2 0.152322703 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=295/9985, ttl=54
 3 0.308649017 OnplusT_55:6d:66 → Broadcast ARP 60 Who has 192.168.0.1? Tell 192.168.0.7
 4 1.002010689 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=296/10241, ttl=64
 5 1.192372266 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=296/10241, ttl=54
 6 2.004542084 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=297/10497, ttl=64
 7 2.183364817 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=297/10497, ttl=54
 8 3.006121371 192.168.0.6 → 104.28.6.89 ICMP 98 Echo (ping) request id=0x1b86, seq=298/10753, ttl=64
 9 3.160247908 104.28.6.89 → 192.168.0.6 ICMP 98 Echo (ping) reply id=0x1b86, seq=298/10753, ttl=54
10 3.632094753 192.168.0.5 → 239.255.255.250 IGMPv2 60 Membership Report group 239.255.255.250
```

## 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
```

```

root@kali:~# tshark -r packets.pcap -V
Running as user "root" and group "root". This could be dangerous.
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
  Interface id: 0 (eth0)
    Interface name: eth0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan 28, 2020 11:38:48.841685525 EST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1580229528.841685525 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 98 bytes (784 bits)
    Capture Length: 98 bytes (784 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:icmp:data]
  Ethernet II, Src: D-LinkIn_59:e1:24 (1c:5f:2b:59:e1:24), Dst: Vmwa
    Destination: Vmware_10:c6:1b (00:0c:29:10:c6:1b)
      Address: Vmware_10:c6:1b (00:0c:29:10:c6:1b)
        .... ..0. .... = LG bit: Globally unique ad
        .... ...0 .... = IG bit: Individual address
    Source: D-LinkIn_59:e1:24 (1c:5f:2b:59:e1:24)
      Address: D-LinkIn_59:e1:24 (1c:5f:2b:59:e1:24)
        .... ..0. .... = LG bit: Globally unique ad
        .... ...0 .... = IG bit: Individual address
    Type: IPv4 (0x0800)
  Internet Protocol Version 4, Src: 104.28.6.89, Dst: 192.168.0.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capa
    Total Length: 84
    Identification: 0x1f72 (8050)
    Flags: 0x0000
      0 ... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
      ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 54

```

## Формат выходных данных

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
```

```
root@kali:~# tshark -T x
Running as user "root" and group "root". This could be dangerous.
tshark: Invalid -T parameter "x"; it must be one of:
  "fields"    The values of fields specified with the -e option, in a form
               specified by the -E option.
  "pdml"      Packet Details Markup Language, an XML-based format for the
               details of a decoded packet. This information is equivalent to
               the packet details printed with the -V flag.
  "ps"        PostScript for a human-readable one-line summary of each of
               the packets, or a multi-line view of the details of each of
               the packets, depending on whether the -V flag was specified.
  "psml"      Packet Summary Markup Language, an XML-based format for the
               summary information of a decoded packet. This information is
               equivalent to the information shown in the one-line summary
               printed by default.
  "json"      Packet Summary, an JSON-based format for the details
               summary information of a decoded packet. This information is
               equivalent to the packet details printed with the -V flag.
  "jsonraw"   Packet Details, a JSON-based format for machine parsing
               including only raw hex decoded fields (same as -T json -x but
               without text decoding, only raw fields included).
  "ek"        Packet Details, an EK JSON-based format for the bulk insert
               into elastic search cluster. This information is
               equivalent to the packet details printed with the -V flag.
  "text"      Text of a human-readable one-line summary of each of the
               packets, or a multi-line view of the details of each of the
               packets, depending on whether the -V flag was specified.
               This is the default.
  "tabs"      Similar to the text report except that each column of the
               human-readable one-line summary is delimited with an ASCII
               horizontal tab character.
```

## 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"?>

```



```
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.
%
```

## 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
```



```
root@kali:~# tshark -r packets.pcap -T psml
Running as user "root" and group "root". This could be dangerous.
<?xml version="1.0" encoding="utf-8"?>
<psml version="0" creator="wireshark/3.0.5">
<structure>
<section>No.</section>
<section>Time</section>
<section>Source</section>
<section>Destination</section>
<section>Protocol</section>
<section>Length</section>
<section>Info</section>
</structure>

<packet>
<section>1</section>
<section>0.000000000</section>
<section>192.168.0.6</section>
<section>104.28.6.89</section>
<section>ICMP</section>
<section>98</section>
<section>Echo (ping) request id=0x1b86, seq=1541/1286, ttl=64</section>
</packet>

<packet>
<section>2</section>
<section>0.209711164</section>
<section>104.28.6.89</section>
<section>192.168.0.6</section>
```

## 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
```

```
root@kali:~# tshark -r packets.pcap -T json
Running as user "root" and group "root". This could be dangerous.
[
  {
    "_index": "packets-2020-01-28",
    "_type": "pcap_file",
    "_score": null,
    "_source": {
      "layers": {
        "frame": {
          "frame.interface_id": "0",
          "frame.interface_id_tree": {
            "frame.interface_name": "eth0"
          },
          "frame.encap_type": "1",
          "frame.time": "Jan 28, 2020 11:57:55.786675361 EST",
          "frame.offset_shift": "0.000000000",
          "frame.time_epoch": "1580230675.786675361",
          "frame.time_delta": "0.000000000",
          "frame.time_delta_displayed": "0.000000000",
          "frame.time_relative": "0.000000000",
          "frame.number": "1",
          "frame.len": "98",
          "frame.cap_len": "98",
          "frame.marked": "0",
          "frame.ignored": "0",
          "frame.protocols": "eth:ethertype:ip:icmp:data"
        },
        "eth": {
          "eth.dst": "1c:5f:2b:59:e1:24",
          "eth.dst_tree": {
```

## ЕК

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 ек
```

```

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

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)

```

## Таблица

Этот формат очень похож на текст, за исключением того, что он включает в себя горизонтальное разделение ASCII (охо9) 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=0x1b86, seq=1541/1286, ttl=64
  2  0.209711164  104.28.6.89  →    192.168.0.6    ICMP    98    Echo (ping) reply
id=0x1b86, 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=0x1b86, seq=1542/1542, ttl=64
  4  1.192770973  104.28.6.89  →    192.168.0.6    ICMP    98    Echo (ping) reply
id=0x1b86, 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=0x1b86, seq=1543/1798, ttl=64
  6  2.434560259  104.28.6.89  →    192.168.0.6    ICMP    98    Echo (ping) reply
id=0x1b86, 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=0x1b86, seq=1544/2054, ttl=64
  8  3.347729784  104.28.6.89  →    192.168.0.6    ICMP    98    Echo (ping) reply
id=0x1b86, 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=0x1b86, seq=1545/2310, ttl=64
 10  4.163455725  104.28.6.89  →    192.168.0.6    ICMP    98    Echo (ping) reply
id=0x1b86, seq=1545/2310, ttl=54 (request in 9)
```

## Разница между декодированными пакетами и кодированными пакетами

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.

# Преобразование PDML-файла HTML-страницы

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:

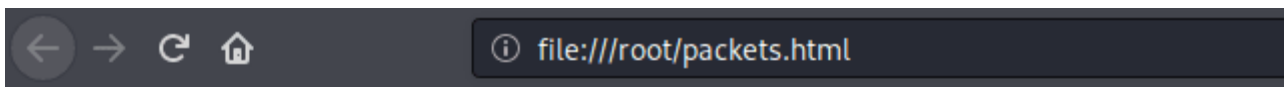
```
xsltproc /usr/share/wireshark/pdml2html.xsl packets.xml > packets.html
```

```
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 &
```





Capture Filename: **packets.pcap** PDML created: **Tue Jan 28 12:03:38 2020**

```
⇒ Frame 1: frame, eth, ip, icmp, [X]
⇒ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
⇒ Ethernet II, Src: Vmware 10:c6:1b (00:0c:29:10:c6:1b), Dst: D-LinkIn_59:e1:24 (1c:5f:2b:
⇒ Internet Protocol Version 4, Src: 192.168.0.6, Dst: 104.28.6.89
⇒ Internet Control Message Protocol
⇒ Frame 2: frame, eth, ip, icmp, [X]
⇒ Frame 3: frame, eth, ip, icmp, [X]
⇒ Frame 4: frame, eth, ip, icmp, [X]
⇒ Frame 5: frame, eth, ip, icmp, [X]
⇒ Frame 6: frame, eth, ip, icmp, [X]
⇒ Frame 7: frame, eth, ip, icmp, [X]
⇒ Frame 8: frame, eth, ip, icmp, [X]
⇒ Frame 9: frame, eth, ip, icmp, [X]
⇒ Frame 10: frame, eth, ip, icmp, [X]
```

## Захват пакетов определенного порта

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

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 HTTP/1.1
 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 HTTP/1.1
 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 HTTP/1.1
 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 HTTP/1.1
```

## Статистические данные

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,bimap
  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
```

## Статистика иерархии протоколов

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
```

## Анализ фильтра для чтения

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
```



```

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
=====

```

## Анализ конечных точек

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

## Анализ бесед

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

```

## Анализ в экспертном режиме

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
```

```

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

```

## Распределение пакетов

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%	

## Длина пакетов

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%

# Анализ выходных данных на основе цвета

**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
```



```
root@kali:~# tshark -r color.pcap --color
```

```
Running as user "root" and group "root". This could be dangerous.
```

```
1 0.000000000 192.168.0.6 → 224.0.0.252 IGMPv2 60 Membership Report group 2
2 1.308991630 192.168.0.137 → 8.8.8.8 DNS 84 Standard query 0xbd04 A det
3 1.309108098 192.168.0.137 → 8.8.8.8 DNS 84 Standard query 0xd70e AAAA
4 1.314734876 8.8.8.8 → 192.168.0.137 DNS 248 Standard query response 0x
cd.akamai.net A 23.32.28.31 A 23.32.28.42
5 1.317061896 8.8.8.8 → 192.168.0.137 DNS 272 Standard query response 0x
.dscd.akamai.net AAAA 2600:140f:3400::1720:1c1f AAAA 2600:140f:3400::1720:1c2a
6 1.351099604 192.168.0.137 → 23.32.28.31 TCP 74 33274 → 80 [SYN] Seq=0 Win=
7 1.360371655 23.32.28.31 → 192.168.0.137 TCP 74 80 → 33274 [SYN, ACK] Seq=0
8 1.360407102 192.168.0.137 → 23.32.28.31 TCP 66 33274 → 80 [ACK] Seq=1 Ack=
9 1.360667272 192.168.0.137 → 23.32.28.31 HTTP 354 GET /success.txt HTTP/1.1
10 1.366231541 23.32.28.31 → 192.168.0.137 TCP 66 80 → 33274 [ACK] Seq=1 Ack=
11 1.368532386 23.32.28.31 → 192.168.0.137 HTTP 473 HTTP/1.1 200 OK (text/pl
12 1.368576332 192.168.0.137 → 23.32.28.31 TCP 66 33274 → 80 [ACK] Seq=289 Ac
13 1.714041355 192.168.0.137 → 8.8.8.8 DNS 72 Standard query 0x2172 A www
14 1.714151234 192.168.0.137 → 8.8.8.8 DNS 72 Standard query 0x6d77 AAAA
15 1.715114796 192.168.0.137 → 8.8.8.8 DNS 73 Standard query 0x3b2e A kal
16 1.715179313 192.168.0.137 → 8.8.8.8 DNS 73 Standard query 0xaf30 AAAA
17 1.715291271 192.168.0.137 → 8.8.8.8 DNS 74 Standard query 0x99d0 A too
18 1.715336702 192.168.0.137 → 8.8.8.8 DNS 74 Standard query 0xa3d2 AAAA
19 1.726762319 8.8.8.8 → 192.168.0.137 DNS 132 Standard query response 0x
20 1.730538887 8.8.8.8 → 192.168.0.137 DNS 133 Standard query response 0x
21 1.780500105 192.168.0.137 → 8.8.8.8 DNS 84 Standard query 0x97f4 A sni
22 1.780608110 192.168.0.137 → 8.8.8.8 DNS 84 Standard query 0x39f9 AAAA
23 1.786609781 8.8.8.8 → 192.168.0.137 DNS 191 Standard query response 0x
24 1.786635886 8.8.8.8 → 192.168.0.137 DNS 211 Standard query response 0x
25 1.790627044 192.168.0.137 → 13.33.169.121 TCP 74 38962 → 443 [SYN] Seq=0 Wi
26 1.833760308 13.33.169.121 → 192.168.0.137 TCP 74 443 → 38962 [SYN, ACK] Seq
27 1.833783843 192.168.0.137 → 13.33.169.121 TCP 66 38962 → 443 [ACK] Seq=1 Ac
```

## Анализ циклического буфера

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

```
root@kali:~# cd packet/
root@kali:~/packet# tshark -i eth0 -w packetsbuffer.pcap -b filesize:1 -b files:3
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
353 ^C

File  Actions  Edit  View  Help

root@kali:~# cd packet/
root@kali:~/packet# ls
packetsbuffer_00009_20200203122531.pcap  packetsbuffer_00010_20200203122531.pcap
root@kali:~/packet# ls -la
total 20
-rwxr-xr-x  2 root root 4096 Feb  3 12:25 .
-rwxr-xr-x 19 root root 4096 Feb  3 12:20 ..
-rw-----  1 root root 1028 Feb  3 12:25 packetsbuffer_00043_20200203122549.pcap
-rw-----  1 root root 1084 Feb  3 12:25 packetsbuffer_00044_20200203122549.pcap
-rw-----  1 root root  252 Feb  3 12:25 packetsbuffer_00045_20200203122549.pcap
root@kali:~/packet# ls -la
total 20
-rwxr-xr-x  2 root root 4096 Feb  3 12:25 .
-rwxr-xr-x 19 root root 4096 Feb  3 12:20 ..
-rw-----  1 root root 1228 Feb  3 12:25 packetsbuffer_00051_20200203122552.pcap
-rw-----  1 root root 1052 Feb  3 12:25 packetsbuffer_00052_20200203122553.pcap
-rw-----  1 root root  552 Feb  3 12:25 packetsbuffer_00053_20200203122554.pcap
```

# Автоматическая остановка работы

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.

## Длина

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
```

```
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] 5099
 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
```

## Размер файла

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
```

```
root@kali:~# tshark -i eth0 -w 1.pcap -a filesize:1
Running as user "root" and group "root". This could be dangerous
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
```

## Типы каналов для передачи данных

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
```

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

## Сведения о версии

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
```



```
root@kali:~# tshark -v ↩
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)

Copyright 1998-2019 Gerald Combs <gerald@wireshark.org> and contributors.
License GPLv2+: GNU GPL version 2 or later <http://www.gnu.org/licenses/old-lic
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled (64-bit) with libpcap, with POSIX capabilities (Linux), with libnl 3,
with GLib 2.60.6, with zlib 1.2.11, with SMI 0.4.8, with c-ares 1.15.0, with L
5.2.4, with GnuTLS 3.6.9 and PKCS #11 support, with Gcrypt 1.8.5, with MIT
Kerberos, with MaxMind DB resolver, with nghttp2 1.39.2, with LZ4, with Snappy
with libxml2 2.9.4.

Running on Linux 5.3.0-kali2-amd64, with Intel(R) Core(TM) i7-9750H CPU @
2.60GHz (with SSE4.2), with 3934 MB of physical memory, with locale en_US.utf8
with libpcap version 1.9.1 (with TPACKET_V3), with GnuTLS 3.6.10, with Gcrypt
1.8.5, with zlib 1.2.11, binary plugins supported (0 loaded).

Built using gcc 9.2.1 20190909.
```

## Функции отчетности

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
```



```
root@kali:~# tshark -G help
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)

Usage: tshark -G [report]

Glossary table reports:
-G column-formats      dump column format codes and exit
-G decodes             dump "layer type"/"decode as" associations and exit
-G dissector-tables    dump dissector table names, types, and properties
-G elastic-mapping     dump ElasticSearch mapping file
-G fieldcount          dump count of header fields and exit
-G fields              dump fields glossary and exit
-G ftypes              dump field type basic and descriptive names
-G heuristic-decodes    dump heuristic dissector tables
-G plugins             dump installed plugins and exit
-G protocols           dump protocols in registration database and exit
-G values              dump value, range, true/false strings and exit

Preference reports:
-G currentprefs        dump current preferences and exit
-G defaultprefs        dump default preferences and exit
-G folders             dump about:folders
```

## Форматы столбцов

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
```

```

root@kali:~# tshark -G column-formats ↩
Running as user "root" and group "root". This could
%q      802.1Q VLAN id
%Yt     Absolute date, as YYYY-MM-DD, and time
%YDOYt  Absolute date, as YYYY/DOY, and time
%At     Absolute time
%V      Cisco VSAN
%B      Cumulative Bytes
%Cus    Custom
%y      DCE/RPC call (cn_call_id / dg_seqnum)
%Tt     Delta time
%Gt     Delta time displayed
%rd     Dest addr (resolved)
%ud     Dest addr (unresolved)
%rD     Dest port (resolved)
%uD     Dest port (unresolved)
%d      Destination address
%D      Destination port
%a      Expert Info Severity
%I      FW-1 monitor if/direction
%F      Frequency/Channel
%hd     Hardware dest addr
%hs     Hardware src addr
%rhd    Hw dest addr (resolved)
%uhd    Hw dest addr (unresolved)
%rhs    Hw src addr (resolved)
%uhs    Hw src addr (unresolved)
%e      IEEE 802.11 RSSI
%x      IEEE 802.11 TX rate
%f      IP DSCP Value
%i      Information
%rnd    Net dest addr (resolved)
%und    Net dest addr (unresolved)
%rns    Net src addr (resolved)
%uns    Net src addr (unresolved)
%nd     Network dest addr
%ns     Network src addr
%m      Number
%L      Packet length (bytes)
%p      Protocol
%Pt     Relative time

```

## Декодирование

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 «**slap.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.
slap.proc.sout 17      slap
slap.proc.sout 3       slap
slap.proc.sout 6       slap
slap.proc.sout 23      slap
slap.proc.sout 9       slap
slap.proc.sout 48      slap
slap.proc.sout 43      slap
slap.proc.sout 29      slap
slap.proc.sout 4       slap
slap.proc.sout 21      slap
```

## Диссектор таблиц

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
```

```

root@kali:~# tshark -G elastic-mapping | head
Running as user "root" and group "root". This could be dangerous.
{
  "template": "packets-*",
  "settings": {
    "index.mapping.total_fields.limit": 1000000
  },
  "mappings": {
    "pcap_file": {
      "dynamic": false,
      "properties": {
        "timestamp": {

```

# Количество полей

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
```

```
root@kali:~# tshark -G fieldcount ↩
Running as user "root" and group "root". This could be dangerous
There are 214494 header fields registered, of which:
    0 are deregistered
    2522 are protocols
    16070 have the same name as another field

215000 fields were pre-allocated.

The header field table consumes 1679 KiB of memory.
The fields themselves consume 15081 KiB of memory.
```

## Поля

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
```

```
root@kali:~# tshark -G fields | head ↩
Running as user "root" and group "root". This could be dangerous.
P      Short Frame      _ws.short
P      Malformed Packet  _ws.malformed
P      Unreassembled Fragmented Packet _ws.unreassembled
F      Dissector bug     _ws.malformed.dissector_bug      FT_NONE _ws.malformed
F      Reassembly error  _ws.malformed.reassembly      FT_NONE _ws.malformed
F      Malformed Packet (Exception occurred) _ws.malformed.expert      FT_NONE _ws.ma
P      Type Length Mismatch _ws.type_length
F      Trying to fetch X with length Y _ws.type_length.mismatch      FT_NONE _ws.ty
P      Number-String Decoding Error _ws.number_string.decoding_error
F      Failed to decode number from string _ws.number_string.decoding_error.failed
x0
```

## Основные виды

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
FT_OTP ASN.1 object identifier

```

## Эвристическое декодирование

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
```

```
root@kali:~# tshark -G heuristic-decodes
Running as user "root" and group "root".
rtsp    rtp      F
sctp    sip      T
sctp    nbap     T
sctp    jxta     T
udp     xml      F
udp     wol      T
udp     wg       T
udp     waveagent T
udp     wassp     F
udp     udt      T
udp     teredo   F
udp     stun     T
udp     srt      T
udp     sprt     T
udp     skype    F
udp     sip      T
udp     rtps     T
udp     rtp      F
udp     rtcp     T
udp     rpcap    T
udp     rpc      T
udp     rlm      T
udp     rlc-nr   F
udp     rlc-lte F
udp     rlc      F
udp     rftap    T
udp     reload-framing T
udp     reload   T
udp     redbackli T
udp     raknet   T
udp     quic     T
udp     proxy    T
udp     pktgen   T
udp     peekremote T
udp     pdcp-nr F
```

## Плагины

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

```

## Протоколы

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

```

## Значения

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
```

```
root@kali:~# tshark -G values | head ↵
Running as user "root" and group "root". This could be dangerous.
R      ieee1722.subtype      0x0      0x0      IEC 61883/IIDC Format
R      ieee1722.subtype      0x1      0x1      MMA Streams
R      ieee1722.subtype      0x2      0x2      AVTP Audio Format
R      ieee1722.subtype      0x3      0x3      Compressed Video Format
R      ieee1722.subtype      0x4      0x4      Clock Reference Format
R      ieee1722.subtype      0x5      0x5      Time Synchronous Control Format
R      ieee1722.subtype      0x6      0x6      SDI Video Format
R      ieee1722.subtype      0x7      0x7      Raw Video Format
R      ieee1722.subtype      0x8      0x6d     Reserved for future protocols
R      ieee1722.subtype      0x6e     0x6e     AES Encrypted Format Continuous
```

## Предпочтения

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
```

```
root@kali:~# tshark -G currentprefs | head ↵
Running as user "root" and group "root". This could be dangerous.
# Configuration file for Wireshark 3.0.5.
#
# This file is regenerated each time preferences are saved within
# Wireshark. Making manual changes should be safe, however.
# Preferences that have been commented out have not been
# changed from their default value.

##### User Interface #####

# Open a console window (Windows only)
```



# Папки

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
```

```
root@kali:~# tshark -G folders ↵
Running as user "root" and group "root". This could be dangerous.
Temp: /tmp
Personal configuration: /root/.config/wireshark
Global configuration: /usr/share/wireshark
System: /etc
Program: /usr/bin
Personal Plugins: /root/.local/lib/wireshark/plugins/3.0
Global Plugins: /usr/lib/x86_64-linux-gnu/wireshark/plugins/3.0
Personal Lua Plugins: /root/.local/lib/wireshark/plugins
Global Lua Plugins: /usr/lib/x86_64-linux-gnu/wireshark/plugins
Extcap path: /usr/lib/x86_64-linux-gnu/wireshark/extcap
MaxMind database path: /usr/share/GeoIP
MaxMind database path: /var/lib/GeoIP
MaxMind database path: /usr/share/GeoIP
MaxMind database path: /var/lib/GeoIP
```

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.

## Установка

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

```
apt install python3
```

```
root@kali:~# apt install python3 ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython3-stdlib python3-minimal
Suggested packages:
  python3-doc python3-venv
The following packages will be upgraded:
  libpython3-stdlib python3 python3-minimal
3 upgraded, 0 newly installed, 0 to remove and 673 not upgraded
Need to get 119 kB of archives.
After this operation, 1,024 B of additional disk space will be
Do you want to continue? [Y/n] y ↵
Get:1 http://ftp.harukasan.org/kali kali-rolling/main amd64 py
Get:2 http://ftp.harukasan.org/kali kali-rolling/main amd64 py
Get:3 http://ftp.harukasan.org/kali kali-rolling/main amd64 li
Fetched 119 kB in 10s (11.7 kB/s)
```

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

```
apt install python3-pip
```

```
root@kali:~# apt install python3-pip ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libc-dev-bin libc6 libc6-dev libc6-i386 libcrypt-dev libcr
  libpython3.7-stdlib python-pip-whl python3-dev python3-en
  python3-secretstorage python3-setuptools python3-wheel py
Suggested packages:
  glibc-doc libkf5wallet-bin gir1.2-gnomekeyring-1.0 python-
The following NEW packages will be installed:
  libcrypt-dev libcrypt1 libpython3-dev libpython3.7-dev py
  python3-keyrings.alt python3-pip python3-secretstorage py
The following packages will be upgraded:
  libc-dev-bin libc6 libc6-dev libc6-i386 libpython3.7 libpy
  python3.7-minimal
10 upgraded, 15 newly installed, 0 to remove and 663 not up
Need to get 59.3 MB of archives
```

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
```

```
root@kali:~# pip3 install pyshark ↵
Collecting pyshark
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection.VerifiedHTTPSConnection object at 0x7fd92b76d610>: Failed to establish a new connection')': /packages/b9/b0/ef87c71f7937ea8124944b2081210f9df10e47d2faa57d7c30d3e12af064/pysha
  Downloading https://files.pythonhosted.org/packages/b9/b0/ef87c71f7937ea8124944b2081210f9d
-none-any.whl
Collecting py (from pyshark)
  Downloading https://files.pythonhosted.org/packages/99/8d/21e1767c009211a62a8e3067280bfce7
ne-any.whl (83kB)
    100% |████████████████████████████████████████| 92kB 1.5MB/s
Requirement already satisfied: lxml in /usr/lib/python3/dist-packages (from pyshark) (4.4.1)
Installing collected packages: py, pyshark
Successfully installed py-1.8.1 pyshark-0.4.2.9
```

## Захват

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
```

```
root@kali:~# python3 ↵
Python 3.7.6 (default, Jan 19 2020, 22:34:52)
[GCC 9.2.1 20200117] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyshark ↵
>>> capture = pyshark.LiveCapture(interface='eth0') ↵
>>> capture.sniff(timeout=5) ↵
>>> capture ↵
<LiveCapture (9 packets)>
```

# Красивая презентация

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()
```

```
>>> capture[1].pretty_print() ↩
Layer ETH:
  Destination: 1c:5f:2b:59:e1:24
  Address: 1c:5f:2b:59:e1:24
  .... ..0. .... = LG bit: Globally unique address (factor 2)
  .... ...0 .... = IG bit: Individual address (unicast)
  Source: 00:0c:29:d5:b7:2d
  Type: IPv4 (0x0800)
  Address: 00:0c:29:d5:b7:2d
  .... ..0. .... = LG bit: Globally unique address (factor 2)
  .... ...0 .... = IG bit: Individual address (unicast)
Layer IP:
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  0000 00.. = Differentiated Services Codepoint: Default (0)
  .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport
  Total Length: 52
  Identification: 0x4b7c (19324)
  Flags: 0x4000, Don't fragment
  0 ... .... = Reserved bit: Not set
  .1.. .... = Don't fragment: Set
  ..0. .... = More fragments: Not set
  ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x62cb [validation disabled]
  Header checksum status: Unverified
  Source: 192.168.0.137
  Destination: 13.35.190.40
Layer TCP:
  Source Port: 38820
  Destination Port: 443
  Stream index: 1
  TCP Segment Len: 0
  Sequence number: 1 (relative sequence number)
  Next sequence number: 1 (relative sequence number)
  Acknowledgment number: 1 (relative ack number)
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x010 (ACK)
  000. .... = Reserved: Not set
  ...0 .... = Nonce: Not set
  .... 0... = Congestion Window Reduced (CWR): Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...1 .... = Acknowledgment: Set
  .... .... 0... = Push: Not set
```

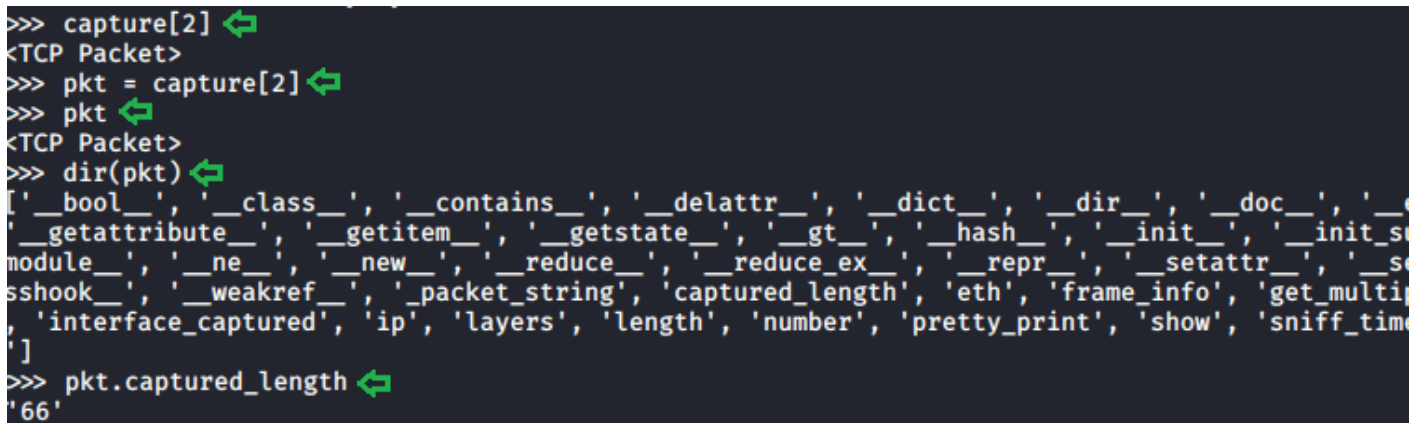
## Захваченное поле длины

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
```



```
>>> capture[2]
<TCP Packet>
>>> pkt = capture[2]
>>> pkt
<TCP Packet>
>>> dir(pkt)
['_bool_', '_class_', '_contains_', '_delattr_', '_dict_', '_dir_', '_doc_', '_getattribute_', '_getitem_', '_getstate_', '_gt_', '_hash_', '_init_', '_init_subclass_', '_le_', '_len_', '_module_', '_ne_', '_new_', '_reduce_', '_reduce_ex_', '_repr_', '_setattr_', '_sizeof_', '_str_', '_subclasshook_', '_weakref_', '_packet_string_', 'captured_length', 'eth', 'frame_info', 'get_multi...', 'interface_captured', 'ip', 'layers', 'length', 'number', 'pretty_print', 'show', 'sniff_time...']
>>> pkt.captured_length
'66'
```

## Слои, поля Src и 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
```

```
>>> pkt.layers
[<ETH Layer>, <IP Layer>, <TCP Layer>]
>>> dir(pkt.eth)
['DATA_LAYER', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__str__', '__subclasshook__', '_field_prefix', '_get_all_field_lines', '_get_all_fields_with_alternates', '_get_all_fields_with_defaults', '_sanitize_field_name', 'addr', 'addr_resolved', 'dst', 'dst_resolved', 'field_name', 'field_value', 'ig', 'layer_name', 'lg', 'pretty_print', 'raw_mode', 'src', 'src_resolved']
>>> pkt.eth.src
'1c:5f:2b:59:e1:24'
>>> pkt.eth.dst
'00:0c:29:d5:b7:2d'
>>> pkt.eth.type
'0x00000800'
```

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()
```

```

>>> dir(pkt.ip)
['DATA_LAYER', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__...', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le...', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__...', '__field_prefix__', '__get_all_field_lines__', '__get_all_fields_with_alternates__', '__get...', '__sanitize_field_name__', 'addr', 'checksum', 'checksum_status', 'dsfield', 'dsfield_d...', 'flags', 'flags_df', 'flags_mf', 'flags_rb', 'frag_offset', 'get', 'get_field', 'get...', 'id', 'layer_name', 'len', 'pretty_print', 'proto', 'raw_mode', 'src', 'src_host',
>>> pkt.ip.src
'13.35.190.40'
>>> pkt.ip.dst
'192.168.0.137'
>>> pkt.ip.pretty_print()
Layer IP:
 0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
0001 00.. = Differentiated Services Codepoint: Unknown (4)
  .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 52
Identification: 0x2e26 (11814)
Flags: 0x4000, Don't fragment
0 ... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 248
Protocol: TCP (6)
Header checksum: 0xc810 [validation disabled]
Header checksum status: Unverified
Source: 13.35.190.40
Destination: 192.168.0.137

```

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

## Захват 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 sniffing 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
```

```
root@kali:~# ifconfig eth0 promisc ↩
root@kali:~# ifconfig eth0 ↩
eth0: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 1500
    inet 192.168.0.137 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::20c:29ff:fed5:b72d prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d5:b7:2d txqueuelen 1000 (Ethernet)
    RX packets 67816 bytes 85545596 (81.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30726 bytes 2463013 (2.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~# tshark -i eth0 -c 10 ↩
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
  1 0.000000000 192.168.0.137 → 35.169.2.62 TLSv1.2 164 Application Data
  2 0.000142943 192.168.0.137 → 107.23.176.98 TLSv1.2 164 Application Data
  3 0.236904732 35.169.2.62 → 192.168.0.137 TLSv1.2 187 Application Data
  4 0.236921665 192.168.0.137 → 35.169.2.62 TCP 66 40520 → 443 [ACK] Seq=99 Ack=122 W
  5 0.242952531 107.23.176.98 → 192.168.0.137 TLSv1.2 187 Application Data
  6 0.242967301 192.168.0.137 → 107.23.176.98 TCP 66 41152 → 443 [ACK] Seq=99 Ack=122 W
  7 1.343354460 192.168.0.6 → 224.0.0.251 IGMPv2 60 Membership Report group 224.0.0.
  8 2.842606464 192.168.0.6 → 224.0.0.252 IGMPv2 60 Membership Report group 224.0.0.
  9 6.807673972 192.168.0.137 → 34.213.241.62 TCP 66 51094 → 443 [ACK] Seq=1 Ack=1 Win
 10 7.100843807 34.213.241.62 → 192.168.0.137 TCP 66 [TCP ACKed unseen segment] 443 →
10 packets captured
root@kali:~# tshark -i eth0 -c 10 -p ↩
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
  1 0.000000000 34.213.241.62 → 192.168.0.137 TLSv1.2 97 Encrypted Alert
  2 0.000019158 192.168.0.137 → 34.213.241.62 TCP 66 51094 → 443 [ACK] Seq=1 Ack=32 Wi
  3 0.000222027 192.168.0.137 → 34.213.241.62 TLSv1.2 97 Encrypted Alert
  4 0.000288786 192.168.0.137 → 34.213.241.62 TCP 66 51094 → 443 [FIN, ACK] Seq=32 Ack
  5 0.289883135 34.213.241.62 → 192.168.0.137 TCP 66 [TCP Previous segment not capture
  6 0.289903932 34.213.241.62 → 192.168.0.137 TCP 66 [TCP Out-Of-Order] 443 → 51094 [F
  7 0.289914338 192.168.0.137 → 34.213.241.62 TCP 66 51094 → 443 [ACK] Seq=33 Ack=33 W
  8 4.120921966 192.168.0.137 → 35.169.2.62 TLSv1.2 165 Application Data
  9 4.121065015 192.168.0.137 → 107.23.176.98 TLSv1.2 164 Application Data
 10 4.394954971 35.169.2.62 → 192.168.0.137 TLSv1.2 188 Application Data
10 packets captured
```

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

# Список используемых источников

- [Руководство по TShark](#)