一、前言

tshark作为wireshark的配套子命令，在CLI场景下能胜任wireshark绝大部分功能，对于需要批量处理、筛选过滤、导出、统计分析、定制化输出等能力。

当然，tshark同时具备抓包和分析包的能力，简单理解为tshark是wireshark的CLI版本，但Linux端抓包基本上业界都会使用tcpdump，并且它基本能覆盖抓包的所有场景，如果想要了解tcpdump的用法，可以参照笔者总结的**这篇文章**。

因此本文将不再讲述tshark的抓包用法，而着重讲述tshark配合诸如grep、sed、awk等文本处理工具如何分析过滤报文、批量处理报文等，拿到我们想要的任何报文内容，这些都是图形化wireshark所做不到的，也是CLI与生俱来的优势。

二、各个发行版安装方式

tshark是wireshark的子命令，默认情况下会携带，gentoo是另外，需要使用USE标记，以下是各发行版的安装命令：

发行版	安装命令
CentOS	yum install wireshark -y
Debian/Ubuntu	apt install wireshark -y
Archlinux	pacman -Syu wireshark
Gentoo	USE="tshark" emerge --ask wireshark

如果是Windows客户端，安装wireshark即可，tshark默认会被安装到wireshark所在路径。

三、用法案例及参数说明

1.读取报文文件不做任何过滤（-r|--read-file）

使用-r|--read-file参数读取抓包文件：

```
tshark -r <filename>
```

```
 ⚙ 16:59:48  ~/pkgs  tshark -r syn.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000  192.168.1.5 → 192.168.1.12 TCP 80 45940 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=52472062 TSecr=0 WS=128
    2   0.000102 192.168.1.12 → 192.168.1.5  TCP 80 8080 → 45940 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1110504068 TSecr=52472062 WS=128
    3   0.000785  192.168.1.5 → 192.168.1.12 TCP 72 45940 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=52472063 TSecr=1110504068
    4   1.837447  192.168.1.5 → 192.168.1.12 TCP 72 45940 → 8080 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=52473900 TSecr=1110504068
    5   1.837951 192.168.1.12 → 192.168.1.5  TCP 72 8080 → 45940 [FIN, ACK] Seq=1 Ack=2 Win=65280 Len=0 TSval=1110505906 TSecr=52473900
    6   1.838621  192.168.1.5 → 192.168.1.12 TCP 72 45940 → 8080 [ACK] Seq=2 Ack=2 Win=64256 Len=0 TSval=52473901 TSecr=1110505906
 ⚙ 16:59:50  ~/pkgs  _
```

会简略的把包文件的报文打印出来，可以看到是一条完整的TCP三次握手与四次挥手的过程。本文着重于如何处理分析已经捕获的报文，因此-r参数在后面的示例中都会被用到。

2.禁止反向解析(-n/-N)

1）禁止一切反向解析（-n）

为了防止IP地址、端口等被反向解析为主机名、端口名时，**-n**参数较为常用，可以更直观看到交互的五元组信息，而不是别名形态：

```
tshark -n -r <filename>
```

```
 ⊙ 17:16:05  ~/pkgs  tshark -n -r http.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⊙ 17:19:10  ~/pkgs
```

2）对传输层端口进行解析（-N t）

区别于-n，**-N**参数则可以做到精准控制解析哪些层级，格式为：`-N <反向解析flag>`，flag取值及含义如下：

flag取值	含义
d	对于DNS包启用解析
m	启用MAC地址解析
n	启用网络地址解析
N	使用外部解析器（例如DNS）进行网络地址解析，n需要被同时启用才有效果
t	启用传输层端口解析
v	启用VLAN ID的名称解析


```
tshark -N t -r <filename>
```

```
 ⊙ 16:06:18  ~/pkgs  tshark -N t -r http.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 http(80) → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → http(80) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 http(80) → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → http(80) [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → http(80) [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 http(80) → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → http(80) [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⊙ 16:06:41  ~/pkgs
```

不难看出，80端口前面加上了**http**协议标识，而**37546**为随机高端口，没有特殊含义，因此没有解析到任何符合特征的协议。

如果要同时启用多个解析，比如同时启用网络地址解析、传输层端口解析，可以是：

```
tshark -N n -N t -r <filename>
```

3.解码输出（-d）

格式为：**-d <layer type>==<selector>,<decode-as protocol>**

也就是wireshark里的“解码为（Decode As）”功能，将特定协议层级，按照手动指定的协议解码输出。

简单理解为，匹配符合条件的包，并将这些包按照指定协议来解析输出一遍。

tshark在默认情况下会根据协议特征和端口标准来自动解码数据包，比如443端口会自动解析为SSL，无需额外指定-d参数，但在某些情况下，特定协议可能有多种变体或扩展，或者端口自定义后并且从协议特征分析有多种协议符合特征，而tshark的自动解码可能只是选择了其中一种，那么手动指定协议解析就派上用场。

执行`tshark -d --help`可以列出支持解码的过滤条件及协议：

```
tshark -d --help|&grep tcp  # 过滤TCP相关的过滤器
```

```
 ⊙ 17:12:42  ~/pkgs  tshark -d --help|&grep tcp
       mbtcp.prot_id (Modbus/TCP protocol identifier)
       rtcp.app.name (RTCP Application Name)
       rtcp.psfb.fmt (RTCP Payload Specific Feedback Message Format)
       rtcp.rtpfb.fmt (RTCP Generic RTP Feedback Message Format)
       tcp.option (TCP Options)
       tcp.port (TCP port)
       tcpcl.v4.sess_ext (TCPCLv4 Session Extension)
       tcpcl.v4.xfer_ext (TCPCLv4 Transfer Extension)
 ⊙ 17:12:46  ~/pkgs
```

因为tshark对于协议的自动解析能力基本上都能覆盖，手上目前还没有这种需要被手动指定协议来解码的场景，但还是可以做一些功能上的演示。

比如将TCP 80端口解码输出为ssh协议：

```
tshark -r <filename> -d 'tcp.port==80,ssh'
```

```
 ⊙ 17:20:22  ~/pkgs  tshark -r http.pcap -d 'tcp.port==80,ssh'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  SSH 146 Client: Encrypted packet (len=74)
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 SSH 335 Server: Encrypted packet (len=263)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⊙ 17:20:34  ~/pkgs
```

这会将符合条件的报文，套用ssh协议来分析解码，再输出到控制台上。

但解码前它实际就是一条正常的HTTP stream：

```
 ⚙ 17:21:42  ~/pkgs  tshark -r http.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⚙ 17:22:44  ~/pkgs  _
```

又或者我们将DNS报文分别解码为rtp、quic协议：

```
tshark -r dns.pcap -d 'udp.port==53,rtp'
tshark -r dns.pcap -d 'udp.port==53,quic'
```

```
 ⚙ 17:24:52  ~/pkgs  tshark -r dns.pcap -d 'udp.port==53,rtp'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.1  DNS 80 Standard query 0x54a4 AAAA api.openai.com
    2   0.000078 192.168.1.12 → 192.168.1.1  DNS 80 Standard query 0x6f3a A api.openai.com
    3   0.000863  192.168.1.1 → 192.168.1.12 DNS 80 Standard query response 0x54a4 AAAA api.openai.com
    4   0.064953  192.168.1.1 → 192.168.1.12 DNS 112 Standard query response 0x6f3a A api.openai.com A 104.18.6.192 A 104.18.7.192
    5   0.232737 192.168.1.12 → 192.168.1.1  DNS 79 Standard query 0xe561 AAAA www.baidu.com
    6   0.233041 192.168.1.12 → 192.168.1.1  RTP 79 PT=Unassigned, SSRC=0x0, Seq=256, Time=65536[Malformed Packet]
    7   0.233391  192.168.1.1 → 192.168.1.12 DNS 109 Standard query response 0xe561 AAAA www.baidu.com CNAME www.a.shifen.com
    8   0.233673  192.168.1.1 → 192.168.1.12 RTP 141 PT=Unassigned, SSRC=0x0, Seq=33152, Time=65539[Malformed Packet]
    9   1.671789 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0x1050 AAAA openwrt.linux-code.com
   10   1.671810 192.168.1.12 → 192.168.1.1  RTP 91 PT=DynamicRTP-Type-103, SSRC=0x0, Seq=256, Time=65536[Malformed Packet]
   11   1.671866 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0xe035 A openwrt.linux-code.com
   12   1.672103 192.168.1.12 → 192.168.1.1  DNS 91 Standard query 0x20fe A prometheus.linux-code.com
   13   1.672641  192.168.1.1 → 192.168.1.12 RTP 91 PT=DynamicRTP-Type-103, SSRC=0x0, Seq=33152, Time=65536[Malformed Packet]
   14   1.672641  192.168.1.1 → 192.168.1.12 DNS 104 Standard query response 0xe035 A openwrt.linux-code.com A 192.168.1.1
   15   1.672641  192.168.1.1 → 192.168.1.12 DNS 107 Standard query response 0x20fe A prometheus.linux-code.com A 192.168.1.12
   16   1.683089  192.168.1.1 → 192.168.1.12 DNS 142 Standard query response 0x1050 AAAA openwrt.linux-code.com CNAME rokas-openwrt.b0.aicdn.com CNAME am.aicdn.com
   17   2.099632 192.168.1.12 → 192.168.1.1  RTCP 88 52862 → 53 Len=40
   18   2.099729 192.168.1.12 → 192.168.1.1  RTP 88 PT=Unassigned, SSRC=0x0, Seq=256, Time=65536[Malformed Packet]
   19   2.100462  192.168.1.1 → 192.168.1.12 RTP 104 PT=Unassigned, SSRC=0x0, Seq=34176, Time=65537[Malformed Packet]
   20   2.100463  192.168.1.1 → 192.168.1.12 RTCP 88 53 → 52862 Len=40
   21   2.640149 192.168.1.12 → 192.168.1.1  DNS 95 Standard query 0x28b9 A qq.com OPT
   22   2.640680  192.168.1.1 → 192.168.1.12 DNS 131 Standard query response 0x28b9 A qq.com A 61.129.7.47 A 183.3.226.35 A 123.151.137.18 OPT
 ⚙ 17:25:00  ~/pkgs  tshark -r dns.pcap -d 'udp.port==53,quic'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.1  QUIC 80 Protected Payload (KP0)
    2   0.000078 192.168.1.12 → 192.168.1.1  QUIC 80 Protected Payload (KP0)
    3   0.000863  192.168.1.1 → 192.168.1.12 QUIC 80 Protected Payload (KP0)
    4   0.064953  192.168.1.1 → 192.168.1.12 QUIC 112 Protected Payload (KP0)
    5   0.232737 192.168.1.12 → 192.168.1.1  QUIC 79 Protected Payload (KP0)
    6   0.233041 192.168.1.12 → 192.168.1.1  QUIC 79 Retry, DCID=00
    7   0.233391  192.168.1.1 → 192.168.1.12 QUIC 109 Protected Payload (KP0)
    8   0.233673  192.168.1.1 → 192.168.1.12 QUIC 141 Retry, DCID=00, SCID=000000
    9   1.671789 192.168.1.12 → 192.168.1.1  QUIC 88 Protected Payload (KP0)
   10   1.671810 192.168.1.12 → 192.168.1.1  QUIC 91 Protected Payload (KP0)
   11   1.671866 192.168.1.12 → 192.168.1.1  QUIC 88 Protected Payload (KP0)
   12   1.672103 192.168.1.12 → 192.168.1.1  QUIC 91 Protected Payload (KP0)
   13   1.672641  192.168.1.1 → 192.168.1.12 QUIC 91 Protected Payload (KP0)
   14   1.672641  192.168.1.1 → 192.168.1.12 QUIC 104 Protected Payload (KP0)
   15   1.672641  192.168.1.1 → 192.168.1.12 QUIC 107 Protected Payload (KP0)
   16   1.683089  192.168.1.1 → 192.168.1.12 QUIC 142 Protected Payload (KP0)
   17   2.099632 192.168.1.12 → 192.168.1.1  QUIC 88 52862 → 53 Len=40[Malformed Packet]
   18   2.099729 192.168.1.12 → 192.168.1.1  QUIC 88 Retry, DCID=00
   19   2.100462  192.168.1.1 → 192.168.1.12 QUIC 104 Retry, DCID=00, SCID=00
   20   2.100463  192.168.1.1 → 192.168.1.12 QUIC 88 53 → 52862 Len=40[Malformed Packet]
   21   2.640149 192.168.1.12 → 192.168.1.1  QUIC 95 Protected Payload (KP0)
   22   2.640680  192.168.1.1 → 192.168.1.12 QUIC 131 Protected Payload (KP0)
 ⚙ 17:25:15  ~/pkgs
```

tshark会将符合特征的报文解码到我们指定的协议，当然这里不能跨协议解析，比如**udp.port==80,http**将udp端口80的包解码成HTTP协议这是不存在的，http是基于TCP的实现，不在一个协议栈里，无法解码。

> 上面的解码仅作为用法示例演示，没有任何意义，请注意辨别。

原始报文为DNS，tshark已经自动帮我们解码适配到正确的协议：

```
 ⚙ 17:25:15  ~/pkgs  tshark -r dns.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.1  DNS 80 Standard query 0x54a4 AAAA api.openai.com
    2   0.000078 192.168.1.12 → 192.168.1.1  DNS 80 Standard query 0x6f3a A api.openai.com
    3   0.000863  192.168.1.1 → 192.168.1.12 DNS 80 Standard query response 0x54a4 AAAA api.openai.com
    4   0.064953  192.168.1.1 → 192.168.1.12 DNS 112 Standard query response 0x6f3a A api.openai.com A 104.18.6.192 A 104.18.7.192
    5   0.232737 192.168.1.12 → 192.168.1.1  DNS 79 Standard query 0xe561 AAAA www.baidu.com
    6   0.233041 192.168.1.12 → 192.168.1.1  DNS 79 Standard query 0xb835 A www.baidu.com
    7   0.233391  192.168.1.1 → 192.168.1.12 DNS 109 Standard query response 0xe561 AAAA www.baidu.com CNAME www.a.shifen.com
    8   0.233673  192.168.1.1 → 192.168.1.12 DNS 141 Standard query response 0xb835 A www.baidu.com CNAME www.a.shifen.com A 14.119.104.189 A 14.119.104.254
    9   1.671789 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0x1050 AAAA openwrt.linux-code.com
   10   1.671810 192.168.1.12 → 192.168.1.1  DNS 91 Standard query 0x9367 AAAA prometheus.linux-code.com
   11   1.671866 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0xe035 A openwrt.linux-code.com
   12   1.672103 192.168.1.12 → 192.168.1.1  DNS 91 Standard query 0x20fe A prometheus.linux-code.com
   13   1.672641  192.168.1.1 → 192.168.1.12 DNS 91 Standard query response 0x9367 AAAA prometheus.linux-code.com
   14   1.672641  192.168.1.1 → 192.168.1.12 DNS 104 Standard query response 0xe035 A openwrt.linux-code.com A 192.168.1.1
   15   1.672641  192.168.1.1 → 192.168.1.12 DNS 107 Standard query response 0x20fe A prometheus.linux-code.com A 192.168.1.12
   16   1.683089  192.168.1.1 → 192.168.1.12 DNS 142 Standard query response 0x1050 AAAA openwrt.linux-code.com CNAME rokas-openwrt.b0.aicdn.com CNAME am.aicdn.com
   17   2.099632 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0x87db AAAA grafana.linux-code.com
   18   2.099729 192.168.1.12 → 192.168.1.1  DNS 88 Standard query 0xb452 A grafana.linux-code.com
   19   2.100462  192.168.1.1 → 192.168.1.12 DNS 104 Standard query response 0xb452 A grafana.linux-code.com A 192.168.1.12
   20   2.100463  192.168.1.1 → 192.168.1.12 DNS 88 Standard query response 0x87db AAAA grafana.linux-code.com
   21   2.640149 192.168.1.12 → 192.168.1.1  DNS 95 Standard query 0x28b9 A qq.com OPT
   22   2.640680  192.168.1.1 → 192.168.1.12 DNS 131 Standard query response 0x28b9 A qq.com A 61.129.7.47 A 183.3.226.35 A 123.151.137.18 OPT
 ⚙ 17:33:53  ~/pkgs
```

4.输出为特定格式（-T）

指定输出格式可以是：**ek|fields|json|jsonraw|pdml|ps|psml|tabs|text**

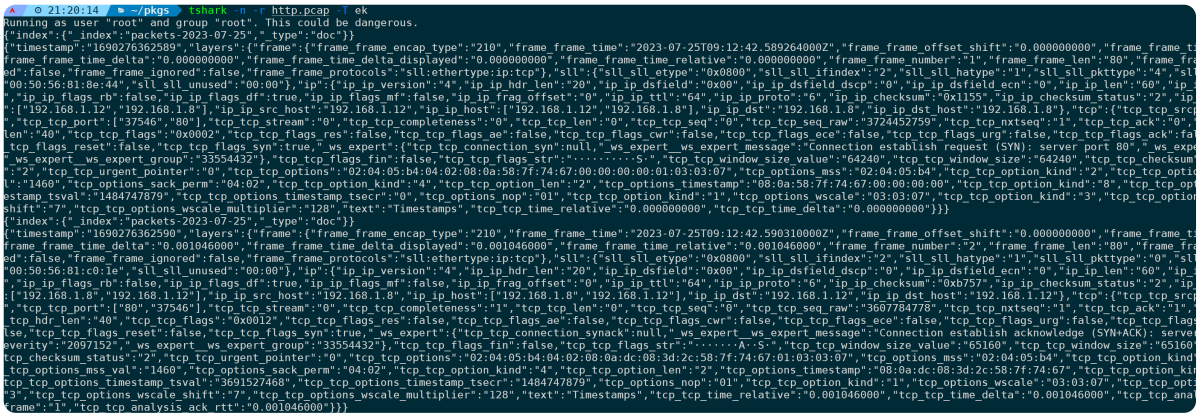
比如输出为json格式可以是：

```
tshark -n -r <filename> -T json
```

```
 21:05:17  ~/pkgs  tshark -n -r http.pcap -T json |more
Running as user "root" and group "root". This could be dangerous.
[
  {
    "_index": "packets-2023-07-25",
    "_type": "doc",
    "_score": null,
    "_source": {
      "layers": {
        "frame": {
          "frame.encap_type": "210",
          "frame.time": "Jul 25, 2023 17:12:42.589264000 CST",
          "frame.offset_shift": "0.000000000",
          "frame.time_epoch": "1690276362.589264000",
          "frame.time_delta": "0.000000000",
          "frame.time_delta_displayed": "0.000000000",
          "frame.time_relative": "0.000000000",
          "frame.number": "1",
          "frame.len": "80",
          "frame.cap_len": "80",
          "frame.marked": "0",
          "frame.ignored": "0",
          "frame.protocols": "sll:ethertype:ip:tcp"
        },
        "sll": {
          "sll.etype": "0x0800",
          "sll.ifindex": "2",
          "sll.hatype": "1",
          "sll.pkttype": "4",
          "sll.halen": "6",
          "sll.src.eth": "00:50:56:81:8e:44",
          "sll.unused": "00:00"
        },
        "ip": {
          "ip.version": "4",
          "ip.hdr_len": "20",
          "ip.dsfield": "0x00",
          "ip.dsfield_tree": {
            "ip.dsfield.dscp": "0",
            "ip.dsfield.ecn": "0"
          },
```

ek格式输出：

```
tshark -n -r <filename> -T ek
```



ek格式：Elasticsearch bulk，代表批量写入Elasticsearch的格式。

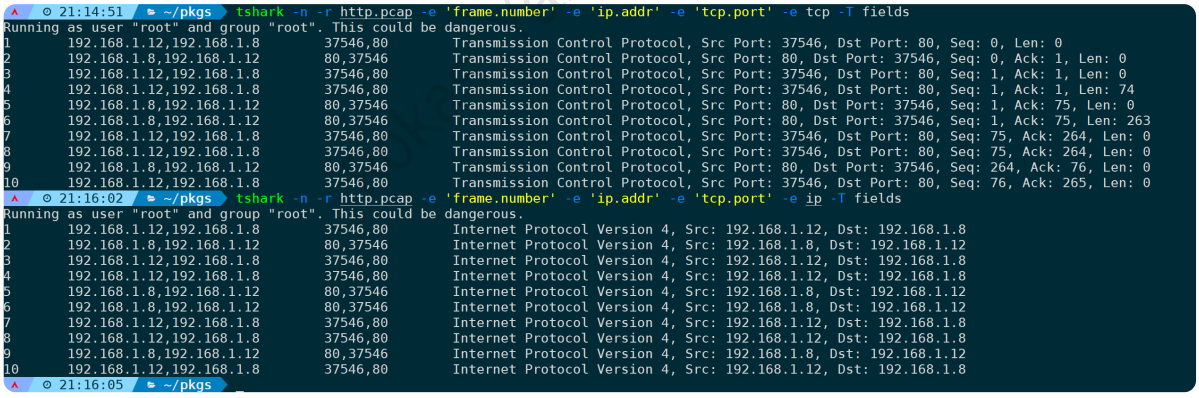
其中**fields**格式需要配合**-e**来使用，不能单独使用，具体用法看下面的**-e**说明。

5.输出报文指定字段（-e）

-e用来指定报文的某个协议字段，给**-T**选项处理输出为最终格式。

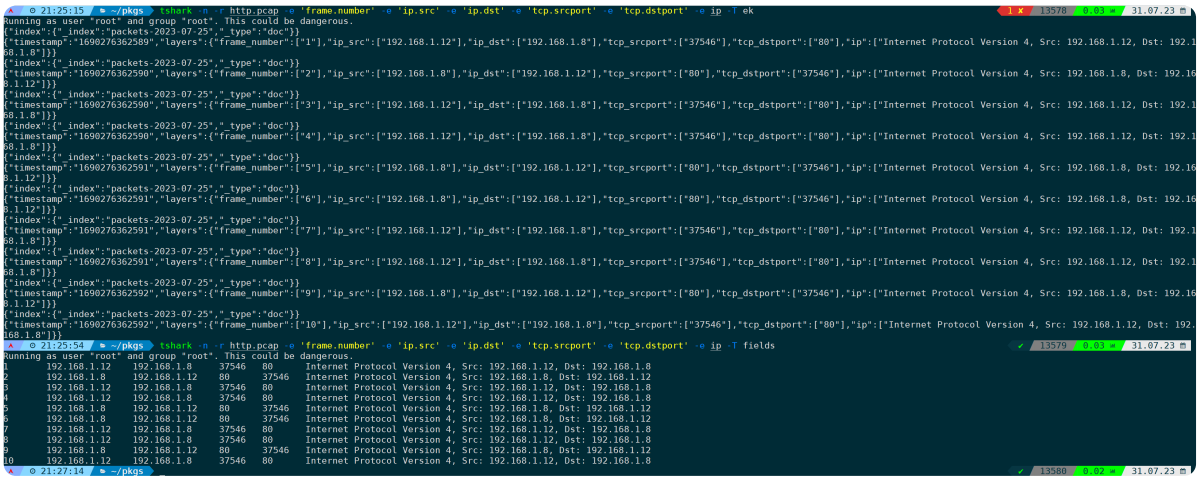
比如，只需要报文帧数、ip地址、端口、tcp或ip协议的字段并最终输出为fields，可以是：

```
tshark -n -r <filename> -e 'frame.number' -e 'ip.addr' -e 'tcp.port' -e tcp -T fields
tshark -n -r <filename> -e 'frame.number' -e 'ip.addr' -e 'tcp.port' -e ip -T fields
```



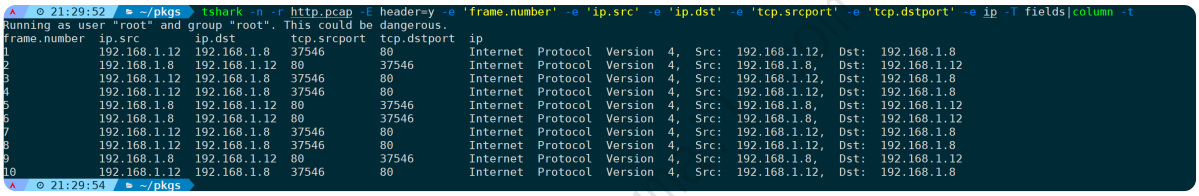
其中，ip.addr、tcp.port都能细分到源ip（ip.src）、目的ip（ip.dst）、tcp源端口（tcp.srcport）、tcp目的端口（tcp.dstport），只要你能想到在wireshark中能作为过滤条件的字段，基本上都可以，用ek格式输出则是：

```
tshark -n -r <filename> -e 'frame.number' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e
'tcp.dstport' -e ip -T ek
```



同时，你还可以加入**-E header=y**参数，来让指定的输出字段，在第一行也对应输出头部字段的含义，之后再通过**column**命令来对齐格式：

```
tshark -n -r <filename> -E header=y -e 'frame.number' -e 'ip.src' -e 'ip.dst' -e
'tcp.srcport' -e 'tcp.dstport' -e ip -T fields|column -t
```



6.设置输出的控制字段（-E）

当通过**-T**参数来输出特定格式时，可以配合**-E**参数来设置一些选项。

-E能接的参数有：

参数选项	默认	含义
bom=y|n	n	在输出前加上UTF-8字节顺序标记（十六进制ef、bb、bf）；
header=y|n	n	打印一个使用-e作为输出第一行的字段名称头部；
separator=/t|/s|<character>	/t	设置字段分隔符；
occurrence=f|l|a	a	打印每个字段的第一次、最后一次或所有出现的内容；
aggregator=,|/s|<character>	,	设置用于每个字段内的分割字符。
quote=d|s|n	n	设置用于环绕字段的引号字符。

下面一一举例。

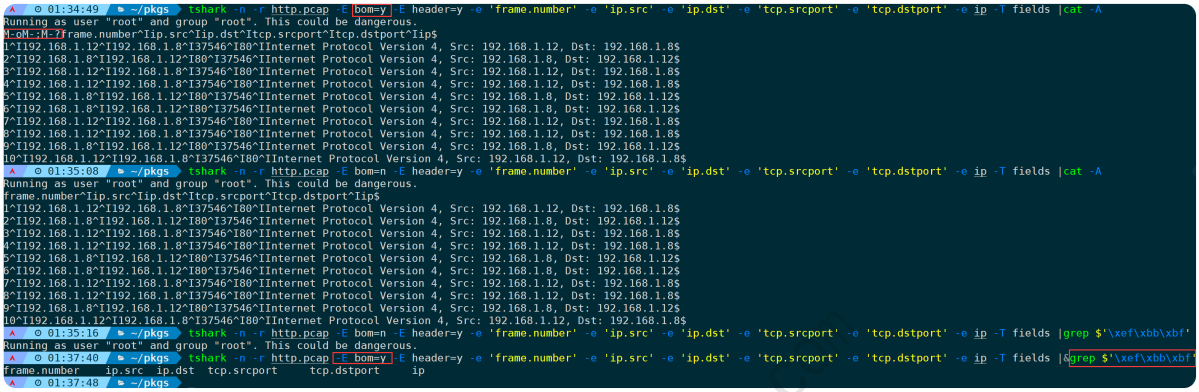
1）bom=y|n

bom（byte-order mark）即字节顺序标记，它是插入到以UTF-8、UTF16或UTF-32编码Unicode文件开头的特殊标记，用来识别Unicode文件的编码类型。对于UTF-8来说，BOM并不是必须的，此选项也不常用，除非有这类需求格式场景。

通过**-E bom=y** 来启用BOM即可，tshark会在控制字段中插入BOM标记。

比如这条命令，我们指定打印输出某些字段，并且通过bom=y参数插入bom格式：

```
tshark -n -r <filename> -E bom=y -E header=y -e 'frame.number' -e 'ip.src' -e 'ip.dst'
-e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
```



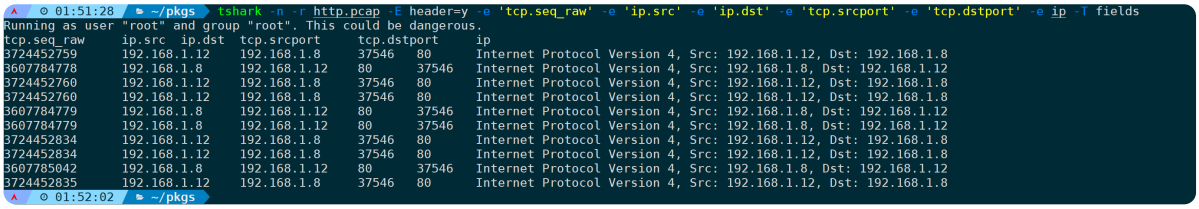
通过对比不难发现，启用bom后，会在输出的开头部分被插入了bom格式，通过**grep -r $``'\xef\xbb\xbf'**也能匹配到此行。

2）header=y

打印一个使用-e作为输出第一行的字段名称头部，前面已有演示，-e指定的字段，将会在第一行也打印出来这些字段名。

比如指定打印tcp绝对序列号以及五元组信息，可以是：

```
tshark -n -r <filename> -E header=y -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e
'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
```



3）separator=/t|/s|<character>

指定字段分隔符，默认为/t，可以指定/s，即单个空格，或者自定义的其它字符。

设置分隔符为两个/t：

```
tshark -n -r <filename> -E separator=/t/t -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
```

设置分隔符为#：

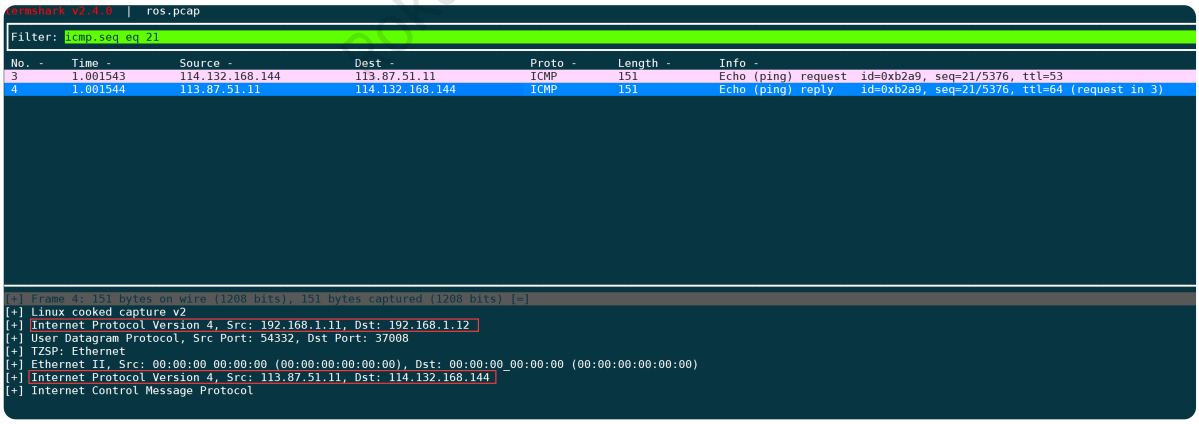
```
tshark -n -r <filename> -E separator=\# -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
```

```
 ⊙ 01:58:44  ~/pkgs  tshark -n -r http.pcap -E separator=\# -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
Running as user "root" and group "root". This could be dangerous.
3724452759#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784778#192.168.1.8#192.168.1.12#80#37546#Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452760#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452760#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784779#192.168.1.8#192.168.1.12#80#37546#Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3607784779#192.168.1.8#192.168.1.12#80#37546#Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452834#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452834#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607785042#192.168.1.8#192.168.1.12#80#37546#Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452835#192.168.1.12#192.168.1.8#37546#80#Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
 ⊙ 01:59:15  ~/pkgs  tshark -n -r http.pcap -E separator=/s -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
Running as user "root" and group "root". This could be dangerous.
3724452759 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784778 192.168.1.8 192.168.1.12 80 37546 Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452760 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452760 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784779 192.168.1.8 192.168.1.12 80 37546 Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3607784779 192.168.1.8 192.168.1.12 80 37546 Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452834 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452834 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607785042 192.168.1.8 192.168.1.12 80 37546 Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452835 192.168.1.12 192.168.1.8 37546 80 Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
 ⊙ 01:59:36  ~/pkgs  tshark -n -r http.pcap -E separator=/t/t -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields
Running as user "root" and group "root". This could be dangerous.
3724452759      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784778      192.168.1.8     192.168.1.12    80      37546   Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452760      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452760      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607784779      192.168.1.8     192.168.1.12    80      37546   Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3607784779      192.168.1.8     192.168.1.12    80      37546   Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452834      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3724452834      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
3607785042      192.168.1.8     192.168.1.12    80      37546   Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.12
3724452835      192.168.1.12    192.168.1.8     37546   80      Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.8
 ⊙ 01:59:46  ~/pkgs
```

4）occurrence=f|l|a

打印每个字段取值的第一个（f）、最后一个（l），所有（a），当一个字段有多个取值时，默认会打印所有取值，可以通过此参数来指定打印第几个。

比如下面这个包，通过IPIP封装，因此ip有内层和外层：



此时我们通过tshark来处理IP字段，occurrence的默认行为是a，即输出字段内所有的内容，会将两个IP都输出出来：

```
tshark -n -r <filename> -E header=y -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
```

```
 ⊙ 02:10:55  ~/pkgs  tshark -n -r ros.pcap -E header=y -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src                    ip.dst
21        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
21        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
 ⊙ 02:11:01  ~/pkgs  tshark -n -r ros.pcap -E occurrence=a -E header=y -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src                    ip.dst
21        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
21        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
 ⊙ 02:11:08  ~/pkgs
```

此时我们只想要内层IP，可以通过指定occurrence=l实现：

```
tshark -n -r <filename> -E header=y -E occurrence=l -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
```

```
 ⚙ 02:05:41  ~/pkgs  tshark -n -r ros.pcap -E header=y -E occurrence=l -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src           ip.dst           ip
21        114.132.168.144  113.87.51.11     Internet  Protocol  Version  4,  Src:  114.132.168.144,  Dst:  113.87.51.11
21        113.87.51.11     114.132.168.144  Internet  Protocol  Version  4,  Src:  113.87.51.11,     Dst:  114.132.168.144
 ⚙ 02:07:14  ~/pkgs
```

只想要外层IP，通过指定occurrence=f实现：

```
tshark -n -r <filename> -E header=y -E occurrence=f -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
```

```
 ⚙ 02:05:34  ~/pkgs  tshark -n -r ros.pcap -E header=y -E occurrence=f -Y 'icmp.seq==21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src        ip.dst        ip
21        192.168.1.11  192.168.1.12  Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12
21        192.168.1.11  192.168.1.12  Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12
 ⚙ 02:05:41  ~/pkgs
```

5）aggregator=,|/s|<character>

指定用于字段内的分隔符，当一个字段有多个取值是，默认会用逗号分割，此参数可用于指定分隔符。

比如将IP层的多个IP字段，用分号分割可以是：

```
tshark -n -r <filename> -E header=y -E aggregator=\; -Y 'icmp.seq>=21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
```

```
 ⚙ 23:54:13  ~/pkgs  tshark -n -r ros.pcap -E header=y -Y 'icmp.seq>=21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src                  ip.dst
21        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
21        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
22        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
22        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
23        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
23        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
24        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
24        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
25        192.168.1.11,114.132.168.144  192.168.1.12,113.87.51.11
25        192.168.1.11,113.87.51.11     192.168.1.12,114.132.168.144
 ⚙ 23:54:20  ~/pkgs  tshark -n -r ros.pcap -E header=y -E aggregator=\; -Y 'icmp.seq>=21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src                  ip.dst
21        192.168.1.11;114.132.168.144  192.168.1.12;113.87.51.11
21        192.168.1.11;113.87.51.11     192.168.1.12;114.132.168.144
22        192.168.1.11;114.132.168.144  192.168.1.12;113.87.51.11
22        192.168.1.11;113.87.51.11     192.168.1.12;114.132.168.144
23        192.168.1.11;114.132.168.144  192.168.1.12;113.87.51.11
23        192.168.1.11;113.87.51.11     192.168.1.12;114.132.168.144
24        192.168.1.11;114.132.168.144  192.168.1.12;113.87.51.11
24        192.168.1.11;113.87.51.11     192.168.1.12;114.132.168.144
25        192.168.1.11;114.132.168.144  192.168.1.12;113.87.51.11
25        192.168.1.11;113.87.51.11     192.168.1.12;114.132.168.144
 ⚙ 23:56:41  ~/pkgs
```

6）quote=d|s|n

设置用于环绕字段的引号字符，**d**表示double，**s**表示single，**n**表示no，默认为**n**，即不使用引号。当输出的字段需要被引号引起来时，可以通过设置此参数实现。

将每个输出字段用双引号引起来，通过**quote=d**来实现：

```
tshark -n -r <filename> -E header=y -E quote=d -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields|column -t
```

```
⊙ 02:39:34 ~/pkgs tshark -n -r http.pcap -E header=y -E quote=d -e 'tcp.seq_raw' -e 'ip.src' -e 'ip.dst' -e 'tcp.srcport' -e 'tcp.dstport' -e ip -T fields|column -t
Running as user "root" and group "root". This could be dangerous.
tcp.seq_raw   ip.src          ip.dst          tcp.srcport  tcp.dstport  ip
"3724452759"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
"3607784778"  "192.168.1.8"   "192.168.1.12"  "80"         "37546"      "Internet  Protocol  Version  4,  Src:  192.168.1.8,   Dst:  192.168.1.12"
"3724452760"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
"3724452760"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
"3607784779"  "192.168.1.8"   "192.168.1.12"  "80"         "37546"      "Internet  Protocol  Version  4,  Src:  192.168.1.8,   Dst:  192.168.1.12"
"3607784779"  "192.168.1.8"   "192.168.1.12"  "80"         "37546"      "Internet  Protocol  Version  4,  Src:  192.168.1.8,   Dst:  192.168.1.12"
"3724452834"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
"3724452834"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
"3607785042"  "192.168.1.8"   "192.168.1.12"  "80"         "37546"      "Internet  Protocol  Version  4,  Src:  192.168.1.8,   Dst:  192.168.1.12"
"3724452835"  "192.168.1.12"  "192.168.1.8"   "37546"      "80"         "Internet  Protocol  Version  4,  Src:  192.168.1.12,  Dst:  192.168.1.8"
⊙ 02:39:38 ~/pkgs
```

quote=s则使用单引号：

```
tshark -n -r <filename> -E header=y -E quote=s -E occurrence=f -Y 'icmp.seq>=21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
```

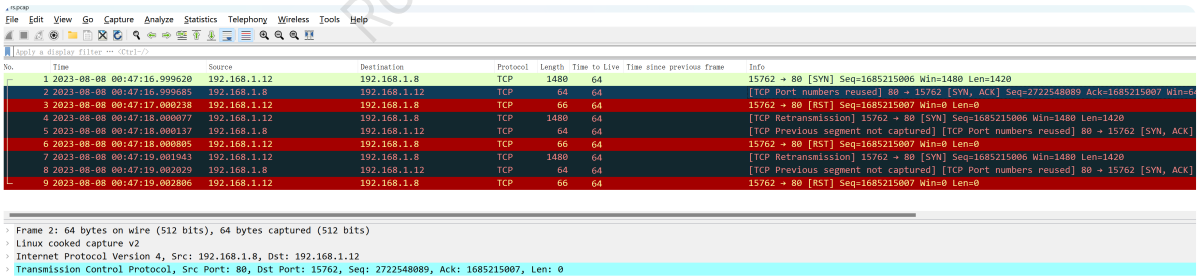
```
⊙ 02:04:03 ~/pkgs tshark -n -r ros.pcap -E header=y -E quote=s -E occurrence=f -Y 'icmp.seq>=21' -e 'icmp.seq' -e 'ip.src' -e 'ip.dst' -e ip -T fields |column -t
Running as user "root" and group "root". This could be dangerous.
icmp.seq  ip.src        ip.dst        ip
'21'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'21'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'22'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'22'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'23'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'23'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'24'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'24'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'25'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
'25'      '192.168.1.11'  '192.168.1.12'  'Internet  Protocol  Version  4,  Src:  192.168.1.11,  Dst:  192.168.1.12'
⊙ 02:04:03 ~/pkgs
```

7.二次依赖分析（-2）

指定此参数，tshark会根据上下文报文的依赖关系（tshark称之为**two-pass**，即进行两次分析），来显示相关报文关联信息，比如：'**response in frame #**'、'**reply in frame**'、'**TCP Port numbers reused**'等字段。

1）端口复用场景（Tcp Port numbers reused）

比如下面这个示例，在wireshark中打开，会显示前后文的依赖关系，比如第2帧提示的端口复用(tcp port numbers reused)：



而如果直接使用tshark来看，是没有这个提示信息的：

```
⊙ 00:52:45 ~/pkgs tshark -n -r rs.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 1480 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    2   0.000065  192.168.1.8 → 192.168.1.12 TCP 64 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    3   0.000618 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
    4   1.000457 192.168.1.12 → 192.168.1.8  TCP 1480 [TCP Retransmission] 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    5   1.000517  192.168.1.8 → 192.168.1.12 TCP 64 [TCP Previous segment not captured] [TCP Port numbers reused] 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    6   1.001185 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
    7   2.002323 192.168.1.12 → 192.168.1.8  TCP 1480 [TCP Retransmission] 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    8   2.002409  192.168.1.8 → 192.168.1.12 TCP 64 [TCP Previous segment not captured] [TCP Port numbers reused] 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    9   2.003186 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
⊙ 00:56:12 ~/pkgs
```

加上-2参数，让它进行一次完整的前后文分析并将这些字段计算出来，则能正常显示：

```
tshark -2 -n -r <filename>
```

```
⊙ 00:56:12 ~/pkgs tshark -2 -n -r rs.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 1480 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    2   0.000065  192.168.1.8 → 192.168.1.12 TCP 64 [TCP Port numbers reused] 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    3   0.000618 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
    4   1.000457 192.168.1.12 → 192.168.1.8  TCP 1480 [TCP Retransmission] 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    5   1.000517  192.168.1.8 → 192.168.1.12 TCP 64 [TCP Previous segment not captured] [TCP Port numbers reused] 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    6   1.001185 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
    7   2.002323 192.168.1.12 → 192.168.1.8  TCP 1480 [TCP Retransmission] 15762 → 80 [SYN] Seq=0 Win=1480 Len=1420
    8   2.002409  192.168.1.8 → 192.168.1.12 TCP 64 [TCP Previous segment not captured] [TCP Port numbers reused] 80 → 15762 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
    9   2.003186 192.168.1.12 → 192.168.1.8  TCP 66 15762 → 80 [RST] Seq=1 Win=0 Len=0
⊙ 00:57:13 ~/pkgs
```

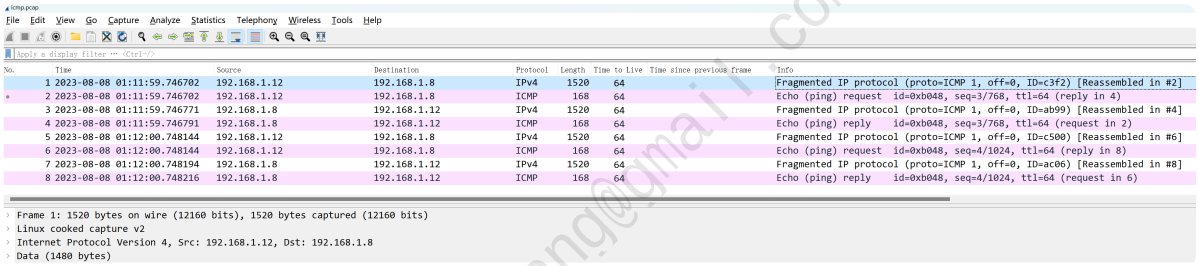
2）ICMP REPLY场景（icmp reply #frame）

再比如icmp场景，不进行二次分析的情况下，第一次的icmp request，并不会在结尾显示**reply in 2**(即reply报文为第二帧)的提示信息：

```
⊙ 01:00:03 ~/pkgs tshark -n -r ros.pcap  -Y 'icmp.seq==20'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 114.132.168.144 → 113.87.51.11 ICMP 151 Echo (ping) request  id=0xb2a9, seq=20/5120, ttl=53
    2   0.000000 113.87.51.11 → 114.132.168.144 ICMP 151 Echo (ping) reply    id=0xb2a9, seq=20/5120, ttl=64 (request in 1)
⊙ 01:00:21 ~/pkgs tshark -n -r ros.pcap  -Y 'icmp.seq==20' -2
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 114.132.168.144 → 113.87.51.11 ICMP 151 Echo (ping) request  id=0xb2a9, seq=20/5120, ttl=53 (reply in 2)
    2   0.000000 113.87.51.11 → 114.132.168.144 ICMP 151 Echo (ping) reply    id=0xb2a9, seq=20/5120, ttl=64 (request in 1)
⊙ 01:02:01 ~/pkgs _
```

3）分片重组场景（Reassembled in #frame）

又或者分片场景，ping一个icmp大包，超过mtu后会分片传输，在wireshark打开后显示如下：



收到的request大包会分成两帧传输，而reply并不需要响应大包，字节数没有超过mtu，因此无需分片。

在tshark中打开则为：

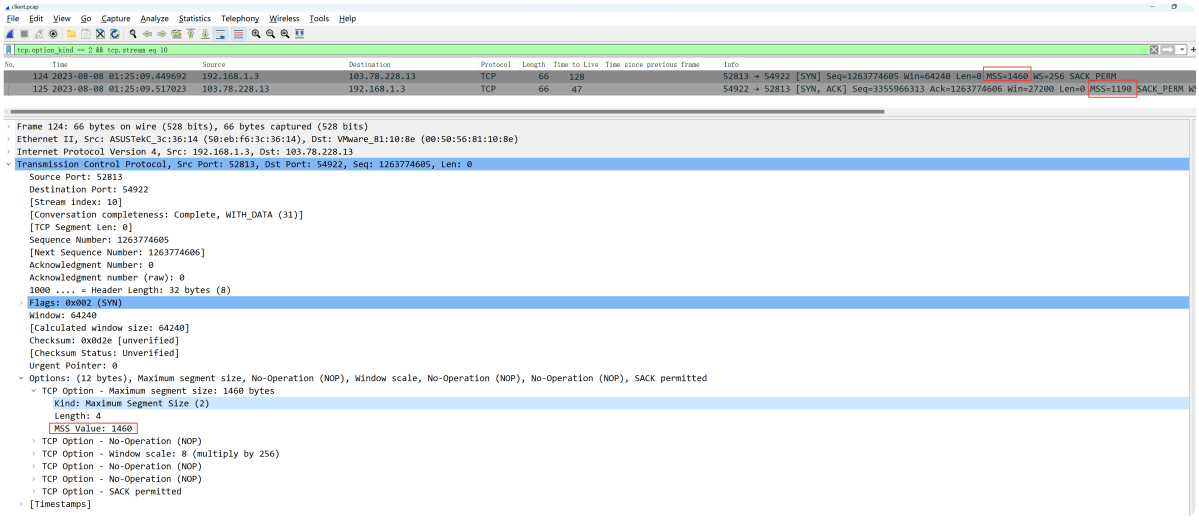
```
⊙ 01:13:24 ~/pkgs tshark -n -r icmp.pcap
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=c3f2)
    2   0.000000 192.168.1.12 → 192.168.1.8  ICMP 168 Echo (ping) request  id=0xb048, seq=3/768, ttl=64
    3   0.000069  192.168.1.8 → 192.168.1.12 IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ab99)
    4   0.000089  192.168.1.8 → 192.168.1.12 ICMP 168 Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
    5   1.001442 192.168.1.12 → 192.168.1.8  IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=c500)
    6   1.001442 192.168.1.12 → 192.168.1.8  ICMP 168 Echo (ping) request  id=0xb048, seq=4/1024, ttl=64
    7   1.001492  192.168.1.8 → 192.168.1.12 IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ac06)
    8   1.001514  192.168.1.8 → 192.168.1.12 ICMP 168 Echo (ping) reply    id=0xb048, seq=4/1024, ttl=64 (request in 6)
⊙ 01:18:29 ~/pkgs _
```

提示有分段，但并没有明确提示哪个段对应哪个request的补充，进行二次分析后则可以明确提示：

```
tshark -n -r <filename> -2
```

```
⊙ 01:18:29 ~/pkgs tshark -n -r icmp.pcap -2
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=c3f2) [Reassembled in #2]
    2   0.000000 192.168.1.12 → 192.168.1.8  ICMP 168 Echo (ping) request  id=0xb048, seq=3/768, ttl=64 (reply in 4)
    3   0.000069  192.168.1.8 → 192.168.1.12 IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ab99) [Reassembled in #4]
    4   0.000089  192.168.1.8 → 192.168.1.12 ICMP 168 Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
    5   1.001442 192.168.1.12 → 192.168.1.8  IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=c500) [Reassembled in #6]
    6   1.001442 192.168.1.12 → 192.168.1.8  ICMP 168 Echo (ping) request  id=0xb048, seq=4/1024, ttl=64 (reply in 8)
    7   1.001492  192.168.1.8 → 192.168.1.12 IPv4 1520 Fragmented IP protocol (proto=ICMP 1, off=0, ID=ac06) [Reassembled in #8]
    8   1.001514  192.168.1.8 → 192.168.1.12 ICMP 168 Echo (ping) reply    id=0xb048, seq=4/1024, ttl=64 (request in 6)
⊙ 01:19:40 ~/pkgs
```

如果是TCP协议，则在三次握手的前两次（SYN和SYN,ACK阶段）两端会协商最小MSS大小（不包括TCP头部+IP头部），这一条TCP流后续的交互包超过MSS则分片传输。



以上只是列举最常见的三个场景，其它场景不一一举例，进行二次分析往往可以更人性化提示前后报文的依赖相关性。

8.设置时间戳格式（-t）

0）总览

支持的格式及说明如下：

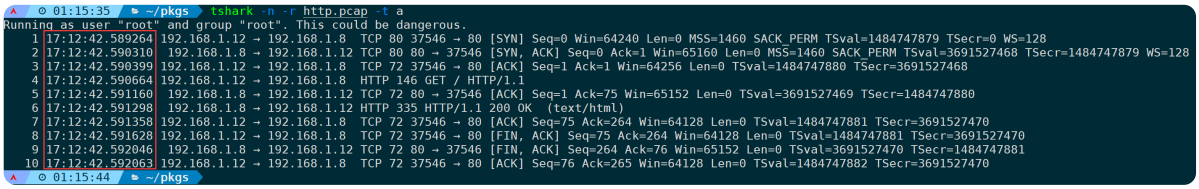
格式	说明
a	绝对时间，抓包的实际时间，不显示日期；
ad	带日期的绝对时间，显示格式为：YYYY-MM-DD；
adoy	年份的绝对天数，显示格式为：YYYY-DAY；
d	delta时间，相对于上一个frame的时间间隔；
dd	delta_displayed时间，相对于上一个已显示的frame的时间间隔；
e	epoch时间/Unix 时间，从1970 年 1 月 1 日 00:00:00（UTC）开始统计的秒数；
r	相对于第一个包的的相对时间；
u	UTC时间，不显示日期；
ud	UTC并且带日期的时间，显示为YYYY-MM-DD；
udoy	UTC时间并且带日期，天数为这一年的绝对天数，显示为：YYYY-DAY。

不指定时默认为r（relative），即相对第一个包的相对时间。

1）当前时区绝对时间（a）

以下面这个包为例，以当前时区的绝对时间显示报文，不显示日期：

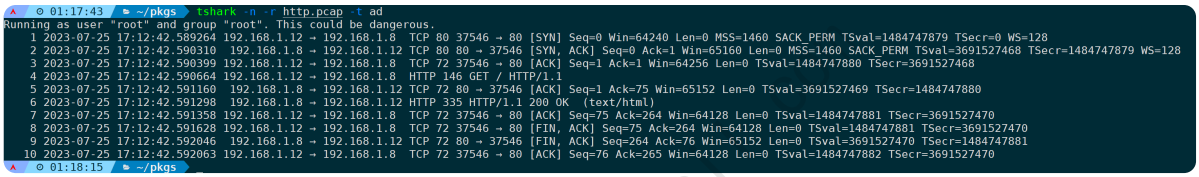
```
tshark -n -r <filename> -t a
```



2）带日期的当前时区绝对时间（ad）

如果需要在a的基础上显示日期，则使用**ad**：

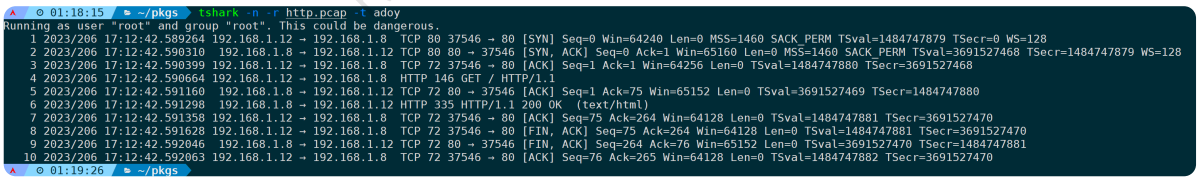
```
tshark -n -r <filename> -t ad
```



3）年份的绝对天数（adoy）

当需要把天数显示为本年度的绝对天数时（Day Of Year，本年的第几天），使用**adoy**：

```
tshark -n -r <filename> -t adoy
```

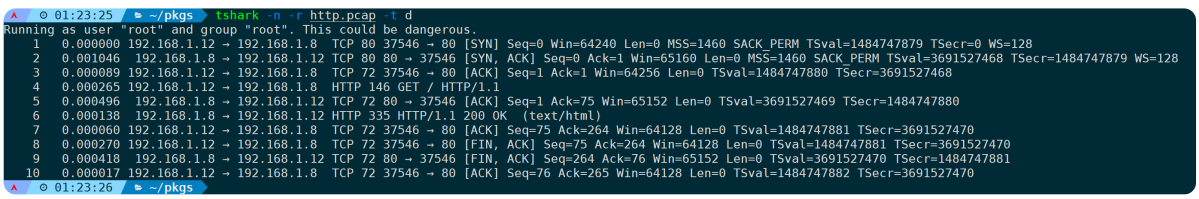


这种场景虽不多见，但存在即合理，方便日期之间做加减。

4）相对于上一个报文的时间间隔（d）

相当于上一个报文的时间间隔，则使用**d**：

```
tshark -n -r <filename> -t d
```



5）相对于上一个已显示的报文时间间隔（dd）

此参数和d的区别是，它的相当时间是已经输出在屏幕上的上一个报文的相对时间。

比如使用dd显示所有报文的时间：

```
tshark -n -r <filename> -t dd
```

```
 ⚙ 01:27:02  ~/pkgs  tshark -n -r http.pcap -t dd
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.000089 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.000265 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.000496  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.000138  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.000060 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.000270 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.000418  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.000017 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⚙ 01:27:29  ~/pkgs
```

这么看，显示的时间和d几乎是一样的，但如果我只过滤HTTP请求，再使用dd：

```
tshark -n -r <filename> -Y 'http' -t dd
```

```
 ⚙ 01:29:45  ~/pkgs  tshark -n -r http.pcap -Y 'http' -t dd
Running as user "root" and group "root". This could be dangerous.
    4   0.000000 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    6   0.000634  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
 ⚙ 01:29:47  ~/pkgs  tshark -n -r http.pcap -Y 'http' -t d
Running as user "root" and group "root". This could be dangerous.
    4   0.000265 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    6   0.000138  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
 ⚙ 01:29:52  ~/pkgs  tshark -n -r http.pcap -t d
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.000089 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.000265 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.000496  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.000138  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.000060 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.000270 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.000418  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.000017 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⚙ 01:30:01  ~/pkgs
```

差别显而易见，当使用HTTP只过滤到两个报文时（frame 4和frame 6）：

- 通过dd输出的时间，frame 4作为输出的第一个包作为时间参考系，所以时间为0，frame 6显示的时间是相对于frame 4的间隔时间；
- 通过d输出的时间，frame 4的时间是相对于frame 3的间隔时间，frame 6的时间相对于frame 5的间隔时间。

所以不难看出，dd的含义是：delta displayed，已经显示了的报文的相对时间。

6）epoch时间/Unix 时间（e）

epoch时间/Unix 时间，从1970 年 1 月 1 日 00:00:00（UTC）开始统计的秒数：

```
tshark -n -r <filename> -t e
```

```
 ⚙ 01:41:50  ~/pkgs  tshark -n -r http.pcap -t e
Running as user "root" and group "root". This could be dangerous.
    1 1690276362.589264 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2 1690276362.590310  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3 1690276362.590399 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4 1690276362.590664 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5 1690276362.591160  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6 1690276362.591298  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7 1690276362.591358 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8 1690276362.591628 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9 1690276362.592046  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10 1690276362.592063 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⚙ 01:42:29  ~/pkgs  TZ='Asia/Shanghai' date -d @1690276362.589264
Tue Jul 25 05:12:42 PM CST 2023
 ⚙ 01:42:32  ~/pkgs
```

7）相对于第一个包的的相对时间（r）

显示相对于第一个包（frame 1）的相对时间，则可以使用r：

```
tshark -n -r <filename> -t r
```

```
 ⊙ 01:44:31  ~/pkgs  tshark -n -r http.pcap -t r
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⊙ 01:44:33  ~/pkgs
```

以最后一个包（frame 10）为例，显示时间间隔为：0.002799，实际上是相对于第一个包的时间，通过capinfos也可以看出首尾包的差距刚好等于这个时间间隔：

```
 ⊙ 01:49:12  ~/pkgs  capinfos http.pcap
File name:           http.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Linux cooked-mode capture v2
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 262144 bytes
Number of packets:   10
File size:           1,257 bytes
Data size:           1,073 bytes
Capture duration:    0.002799 seconds
First packet time:   2023-07-25 17:12:42.589264
Last packet time:    2023-07-25 17:12:42.592063
Data byte rate:      383 kBps
Data bit rate:       3,066 kbps
Average packet size: 107.30 bytes
Average packet rate: 3,572 packets/s
SHA256:              2eda14356b7416db2cd8fcdff869eb5292ea4210cc854689ee7453b5465cdedb
RIPEMD160:           25062ca5c3110f884434c294657c9f8436fc97c6
SHA1:                ee62d2e7f76ce168e7d4e2e784c19981de09300d
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                     Encapsulation = Linux cooked-mode capture v2 (210 - linux-sll2)
                     Capture length = 262144
                     Time precision = microseconds (6)
                     Time ticks per second = 1000000
                     Number of stat entries = 0
                     Number of packets = 10
 ⊙ 01:49:30  ~/pkgs  awk 'BEGIN{print 0.592063-0.589264}'
0.002799
 ⊙ 01:49:33  ~/pkgs  capinfos http.pcap -u
File name:           http.pcap
Capture duration:    0.002799 seconds
 ⊙ 01:49:39  ~/pkgs
```

8）不显示日期的UTC时间（u）

以UTC时间显示，并且不显示日期，则可以使用u：

```
tshark -n -r <filename> -t u
```

```
 ⊙ 01:51:51  ~/pkgs  tshark -n -r http.pcap -t u
Running as user "root" and group "root". This could be dangerous.
    1 09:12:42.589264 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2 09:12:42.590310  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3 09:12:42.590399 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4 09:12:42.590664 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5 09:12:42.591160  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6 09:12:42.591298  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7 09:12:42.591358 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8 09:12:42.591628 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9 09:12:42.592046  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10 09:12:42.592063 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
 ⊙ 01:51:54  ~/pkgs
```

9）显示日期的UTC时间（ud）

在UTC的基础上显示带日期的UTC时间：

```
tshark -n -r <filename> -t ud
```

```
⊙ 01:53:20 ~/pkgs tshark -n -r http.pcap -t ud
Running as user "root" and group "root". This could be dangerous.
    1 2023-07-25 09:12:42.589264 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2 2023-07-25 09:12:42.590310  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3 2023-07-25 09:12:42.590399 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4 2023-07-25 09:12:42.590664 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5 2023-07-25 09:12:42.591160  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6 2023-07-25 09:12:42.591298  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7 2023-07-25 09:12:42.591358 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8 2023-07-25 09:12:42.591628 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9 2023-07-25 09:12:42.592046  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10 2023-07-25 09:12:42.592063 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
⊙ 01:54:06 ~/pkgs
```

10）年度绝对天数的UTC时间（udoy）

显示UTC带日期的时间，其中天数显示为Day Of Year格式，即本年度的第几天：

```
tshark -n -r <filename> -t udoy
```

```
⊙ 01:56:30 ~/pkgs tshark -n -r http.pcap -t udoy
Running as user "root" and group "root". This could be dangerous.
    1 2023/206 09:12:42.589264 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2 2023/206 09:12:42.590310  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3 2023/206 09:12:42.590399 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4 2023/206 09:12:42.590664 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5 2023/206 09:12:42.591160  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6 2023/206 09:12:42.591298  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7 2023/206 09:12:42.591358 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8 2023/206 09:12:42.591628 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9 2023/206 09:12:42.592046  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10 2023/206 09:12:42.592063 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
⊙ 01:56:46 ~/pkgs
```

9.筛选过滤报文（-Y）

此选项常用，用来过滤分析符合过滤表达式的报文，相当于wireshark最上面的过滤筛选栏功能。

上面的一些示例中，其实已经用到了-Y选项来过滤报文，只要你写的过滤表达式符合wireshark语法，就能被正常执行，wireshark语法可以参考 **官方文档**。

1）示例1：通过http.host过滤

想过滤到http host为某个值，可以是：

```
tshark -n -r <filename> -Y 'http.host == "web-server1"'
```

```
⊙ 14:40:50 ~/pkgs tshark -n -r http.pcap -Y 'http.host == "web-server1"'
Running as user "root" and group "root". This could be dangerous.
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
⊙ 14:40:58 ~/pkgs
```

2）示例2：过滤TCP重传、快速重传、DUP ACK的包

过滤TCP重传、快速重传、DUP ACK的包，可以是：

```
tshark -n -r <filename> -Y 'tcp.stream eq 2 && ( tcp.analysis.retransmission or
tcp.analysis.fast_retransmission or tcp.analysis.duplicate_ack )' -t d
```

```
⊙ 14:47:07 ~/pkgs tshark -n -r retransmission.pcap -Y 'tcp.stream eq 2 && ( tcp.analysis.retransmission or tcp.analysis.fast_retransmission or tcp.analysis.duplicate_ack )' -t d
Running as user "root" and group "root". This could be dangerous.
  165   0.115866  192.168.1.3 → 114.67.251.123 TCP 1244 [TCP Retransmission] 58865 → 40616 [PSH, ACK] Seq=3160 Ack=1914 Win=262656 Len=1190
  167   0.000218  192.168.1.3 → 114.67.251.123 TCP 1244 [TCP Retransmission] 58865 → 40616 [PSH, ACK] Seq=551 Ack=1914 Win=262656 Len=1190
  168   0.000050  192.168.1.3 → 114.67.251.123 TCP 1244 [TCP Retransmission] 58865 → 40616 [PSH, ACK] Seq=1741 Ack=1914 Win=262656 Len=1190
  169   0.000000  192.168.1.3 → 114.67.251.123 TCP 283 [TCP Retransmission] 58865 → 40616 [PSH, ACK] Seq=2931 Ack=1914 Win=262656 Len=229
  198   0.200488 114.67.251.123 → 192.168.1.3  TCP 1165 [TCP Spurious Retransmission] 40616 → 58865 [PSH, ACK] Seq=10563 Ack=4350 Win=39936 Len=1111
  199   0.000074  192.168.1.3 → 114.67.251.123 TCP 66 [TCP Dup ACK 195#1] 58865 → 40616 [ACK] Seq=4351 Ack=11674 Win=262912 Len=0 SLE=10563 SRE=11674
  200   0.027773  192.168.1.3 → 114.67.251.123 TCP 54 [TCP Retransmission] 58865 → 40616 [FIN, ACK] Seq=4350 Ack=11674 Win=262912 Len=0
⊙ 14:47:45 ~/pkgs
```

通过-t d来显示相对上一帧的时间，**tcp.stream eq 2**来指定第三条TCP流。

3）示例3：过滤ARP报文

过滤ARP报文，取出特定帧，可以是：

```
tshark -n -r <filename> -Y 'arp && frame.number in {9,10,197,208,216}' -t d
```

```
⊙ 14:52:19 ~/pkgs tshark -n -r retransmission.pcap -Y 'arp && frame.number in {9,10,197,208,216}' -t d
Running as user "root" and group "root". This could be dangerous.
    9   0.283938 f4:6d:2f:a2:a5:12 → ff:ff:ff:ff:ff:ff ARP 60 Who has 192.168.1.3? Tell 192.168.1.200
   10   0.000024 50:eb:f6:3c:36:14 → f4:6d:2f:a2:a5:12 ARP 42 192.168.1.3 is at 50:eb:f6:3c:36:14
  197   0.071903 00:0c:29:db:21:3e → ff:ff:ff:ff:ff:ff ARP 60 Who has 192.168.1.2? Tell 192.168.1.99
  208   0.113325 f4:6d:2f:a2:a5:12 → ff:ff:ff:ff:ff:ff ARP 60 Who has 192.168.1.128? Tell 192.168.1.200
  216   0.189012 00:50:56:81:10:4f → ff:ff:ff:ff:ff:ff ARP 60 Gratuitous ARP for 192.168.1.82 (Request)
⊙ 14:53:02 ~/pkgs
```

4）示例4：过滤第一次握手的请求

第一次握手只发送SYN，ACK位置0，因此可以这么写过滤规则：

```
tshark -n -r http-keep-alive.pcap -Y 'tcp.flags.syn==1&&tcp.flags.ack==0'
```

```
⊙ 20:33:17 ~/pkgs tshark -n -r http-keep-alive.pcap -Y 'tcp.flags.syn==1&&tcp.flags.ack==0'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000  192.168.1.3 → 192.168.1.72 TCP 66 37334 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
    9   0.014748  192.168.1.3 → 192.168.1.72 TCP 66 37335 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   36   7.609547  192.168.1.3 → 192.168.1.72 TCP 66 37351 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   42   7.610776  192.168.1.3 → 192.168.1.72 TCP 66 37352 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   47   7.611210  192.168.1.3 → 192.168.1.72 TCP 66 37353 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   50   7.611476  192.168.1.3 → 192.168.1.72 TCP 66 37354 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  516  40.190998  192.168.1.3 → 192.168.1.81 TCP 66 37462 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
⊙ 20:33:22 ~/pkgs
```

以上只是几个示例，涉及到各个场景的过滤用法，-Y都能完全支持。

10.统计分析报文（-z）

使用`tshark -z help`可以查看支持的统计分析选项。

```
 15:14:36  ~/pkgs  tshark -z help|& awk '{line+=1}END{print line}'
169
 15:14:42  ~/pkgs  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
     asap,stat
     bacapp_instanceid,tree
     bacapp_ip,tree
     bacapp_objectid,tree
     bacapp_service,tree
     calcappprotocol,stat
     camel,counter
     camel,srt
     collectd,tree
     componentstatusprotocol,stat
     conv,bluetooth
     conv,dccp
     conv,eth
     conv,fc
     conv,fddi
     conv,ip
     conv,ipv6
     conv,ipx
     conv,jxta
     conv,mptcp
     conv,ncp
     conv,opensafety
     conv,rsvp
     conv,sctp
     conv,sll
     conv,tcp
     conv,tr
     conv,udp
     conv,usb
     conv,wlan
     conv,wpan
     conv,zbee_nwk
     credentials
     dcerpc,srt
     dests,tree
     dhcp,stat
     diameter,avp
     diameter,srt
     dns,tree
```


目前为止有169个统计分析选项，但常用的并不多。

下面将一一举例最常见的一些示例常见，如果你遇到的需要统计分析的场景并不在举例场景中，那么通过**-z help**去查看相匹配的场景选项即可。

1）统计分析IP会话（conv,ip）

会话统计需要用到**'conv,'**作为前缀，表示的是conversation。

以这一条TCP流为例，统计会话：

```
tshark -n -r <filename> -z 'conv,ip'
```

```
⚙ 15:22:10 ~/pkgs tshark -n -r http.pcap -z 'conv,ip'
Running as user "root" and group "root". This could be dangerous.
    1   0.000000 192.168.1.12 → 192.168.1.8  TCP 80 37546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1484747879 TSecr=0 WS=128
    2   0.001046  192.168.1.8 → 192.168.1.12 TCP 80 80 → 37546 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3691527468 TSecr=1484747879 WS=128
    3   0.001135 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1484747880 TSecr=3691527468
    4   0.001400 192.168.1.12 → 192.168.1.8  HTTP 146 GET / HTTP/1.1
    5   0.001896  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [ACK] Seq=1 Ack=75 Win=65152 Len=0 TSval=3691527469 TSecr=1484747880
    6   0.002034  192.168.1.8 → 192.168.1.12 HTTP 335 HTTP/1.1 200 OK  (text/html)
    7   0.002094 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    8   0.002364 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [FIN, ACK] Seq=75 Ack=264 Win=64128 Len=0 TSval=1484747881 TSecr=3691527470
    9   0.002782  192.168.1.8 → 192.168.1.12 TCP 72 80 → 37546 [FIN, ACK] Seq=264 Ack=76 Win=65152 Len=0 TSval=3691527470 TSecr=1484747881
   10   0.002799 192.168.1.12 → 192.168.1.8  TCP 72 37546 → 80 [ACK] Seq=76 Ack=265 Win=64128 Len=0 TSval=1484747882 TSecr=3691527470
================================================================================
IPv4 Conversations
Filter:<No Filter>
                                               |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                               | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.12        <-> 192.168.1.8                    4 559 bytes       6 514 bytes      10 1,073 bytes     0.000000000         0.0028
================================================================================
⚙ 15:22:38 ~/pkgs
```

可以通过-q参数来让它只输出统计结果，不显示报文：

```
tshark -n -q -r <filename> -z 'conv,ip'
```

```
⚙ 15:22:38 ~/pkgs tshark -n -q -r http.pcap -z 'conv,ip'
Running as user "root" and group "root". This could be dangerous.
================================================================================
IPv4 Conversations
Filter:<No Filter>
                                               |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                               | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.12        <-> 192.168.1.8                    4 559 bytes       6 514 bytes      10 1,073 bytes     0.000000000         0.0028
================================================================================
⚙ 15:24:42 ~/pkgs
```

如果是ipv6地址，改成**-z 'conv,ipv6'**即可。

当然，也可以在统计结果后面来写过滤规则，比如只统计第一条流的结果：

```
tshark -n -q -r <filename> -z conv,ip,tcp.stream==0
```

```
⚙ 15:26:38 ~/pkgs tshark -n -q -r tls.pcap -z conv,ip,tcp.stream==0
Running as user "root" and group "root". This could be dangerous.
================================================================================
IPv4 Conversations
Filter:tcp.stream==0
                                               |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                               | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3         <-> 101.91.37.19                   5 441 bytes       5 1,767 bytes      10 2,208 bytes     0.000000000         0.1060
================================================================================
⚙ 15:26:42 ~/pkgs
```

2）统计分析TCP会话（conv,tcp）

和第一个示例的区别只是统计的层级不一样，**conv,ip** 统计IP层，那么**conv,tcp**则统计tcp头部，既然是TCP层，肯定会有端口存在：

```
tshark -n -q -r <filename> -z conv,tcp
```

```
⚙ 15:29:27 ~/pkgs tshark -n -q -r tls.pcap -z conv,tcp
Running as user "root" and group "root". This could be dangerous.
================================================================================
TCP Conversations
Filter:<No Filter>
                                                  |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                  | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3:58865         <-> 114.67.251.123:40616       31 14 kB          140 15 kB         171 30 kB       1.168876000         1.3720
192.168.1.3:58863         <-> 101.91.37.19:443            5 441 bytes       5 1,767 bytes      10 2,208 bytes     0.000000000         0.1060
================================================================================
⚙ 15:29:33 ~/pkgs
```

当然它也支持过滤选项：

```
tshark -n -q -r <filename> -z conv,tcp,tcp.port==443
```

```
 ⊙ 15:33:53  ~/pkgs  tshark -n -q -r tls.pcap -z conv,tcp,tcp.port==443
Running as user "root" and group "root". This could be dangerous.
================================================================================
TCP Conversations
Filter:tcp.port==443
                                                |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3:58863          <-> 101.91.37.19:443                5 441 bytes       5 1,767 bytes      10 2,208 bytes     0.000000000          0.1060
================================================================================
 ⊙ 15:33:56  ~/pkgs
```

或者，我只想要第二个的流的统计数据：

```
tshark -n -q -r <filename> -z conv,tcp,'tcp.stream eq 2'
```

```
 ⊙ 22:58:29  ~/pkgs  tshark -n -q -r retransmission.pcap -z conv,tcp,'tcp.stream eq 2'
Running as user "root" and group "root". This could be dangerous.
================================================================================
TCP Conversations
Filter:tcp.stream eq 2
                                                |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3:58865          <-> 114.67.251.123:40616           31 14 kB          140 15 kB          171 30 kB         2.143865000          1.3720
================================================================================
 ⊙ 23:00:44  ~/pkgs
```

3）统计分析UDP会话（conv,udp）

通过以上两个示例，你也可以猜到统计命令应该写成下面这样：

```
tshark -n -q -r <filename> -z conv,udp
```

```
 ⊙ 15:38:11  ~/pkgs  tshark -n -q -r retransmission.pcap -z conv,udp
Running as user "root" and group "root". This could be dangerous.
================================================================================
UDP Conversations
Filter:<No Filter>
                                                |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3:4020           <-> 183.47.125.117:8000            10 1,938 bytes       8 904 bytes      18 2,842 bytes     0.401511000          5.5740
fe80::20c:29ff:fe24:8c5c:546 <-> ff02::1:2:547                 0 0 bytes         1 150 bytes       1 150 bytes     3.467620000          0.0000
192.168.1.99:8567          <-> 192.168.1.3:8803                0 0 bytes         1 77 bytes        1 77 bytes      3.513060000          0.0000
================================================================================
 ⊙ 15:39:16  ~/pkgs
```

过滤规则如果有多条或者空格，可以通过单引号引起来：

```
tshark -n -q -r <filename> -z conv,udp,'udp.port in {8000,8803}'
```

```
 ⊙ 15:39:16  ~/pkgs  tshark -n -q -r retransmission.pcap -z conv,udp,'udp.port in {8000,8803}'
Running as user "root" and group "root". This could be dangerous.
================================================================================
UDP Conversations
Filter:udp.port in {8000,8803}
                                                |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.3:4020           <-> 183.47.125.117:8000            10 1,938 bytes       8 904 bytes      18 2,842 bytes     0.401511000          5.5740
192.168.1.99:8567          <-> 192.168.1.3:8803                0 0 bytes         1 77 bytes        1 77 bytes      3.513060000          0.0000
================================================================================
 ⊙ 15:41:07  ~/pkgs
```

4）统计分析DNS层次结构（dns,tree）

分析DNS层级结构信息可以是：

```
tshark -n -q -r <filename> -z dns,tree
```

```
⊙ 15:46:29 ~/pkgs tshark -n -q -r dns.pcap -z dns,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
DNS:
Topic / Item                  Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
Total Packets                 6                                                       0.0013        100%          0.0200        2.585
 rcode                        6                                                       0.0013        100.00%       0.0200        2.585
  No error                    6                                                       0.0013        100.00%       0.0200        2.585
 opcodes                      6                                                       0.0013        100.00%       0.0200        2.585
  Standard query              6                                                       0.0013        100.00%       0.0200        2.585
 Query/Response               6                                                       0.0013        100.00%       0.0200        2.585
  Response                    3                                                       0.0006        50.00%        0.0100        2.586
  Query                       3                                                       0.0006        50.00%        0.0100        2.585
 Query Type                   6                                                       0.0013        100.00%       0.0200        2.585
  A (Host Address)            6                                                       0.0013        100.00%       0.0200        2.585
 Class                        6                                                       0.0013        100.00%       0.0200        2.585
  IN                          6                                                       0.0013        100.00%       0.0200        2.585
Payload size                  6             79.33         47            156           0.0013        100%          0.0200        2.585
Query Stats                   0                                                       0.0000        100%          -             -
 Qname Len                    3             9.33          6             13            0.0006                      0.0100        2.585
 Label Stats                  0                                                       0.0000                      -             -
  2nd Level                   2                                                       0.0004                      0.0100        3.818
  3rd Level                   1                                                       0.0002                      0.0100        2.585
  4th Level or more           0                                                       0.0000                      -             -
  1st Level                   0                                                       0.0000                      -             -
Response Stats                0                                                       0.0000        100%          -             -
 no. of questions             6             1.00          1             1             0.0013                      0.0200        2.586
 no. of authorities           6             0.00          0             0             0.0013                      0.0200        2.586
 no. of answers               6             3.33          2             4             0.0013                      0.0200        2.586
 no. of additionals           6             1.00          1             1             0.0013                      0.0200        2.586
Service Stats                 0                                                       0.0000        100%          -             -
 request-response time (msec) 3             0.50          0.470000      0.571000      0.0006                      0.0100        2.586
 no. of unsolicited responses 0                                                       0.0000                      -             -
 no. of retransmissions       0                                                       0.0000                      -             -

--------------------------------------------------------------------------------------------------------------------------------
⊙ 15:46:34 ~/pkgs
```

将会统计报文中所有设计到dns协议的，响应时间、响应速率、各类查询类别、payload大小、附加记录及各类属于DNS头部字段的统计信息。

当然它也支持过滤选项，比如只统计dns query baidu.com的请求，可以是：

```
tshark -n -q -r <filename> -z dns,tree,dns.'qry.name == baidu.com'
```

```
⊙ 15:53:34 ~/pkgs tshark -n -q -r dns.pcap -z dns,tree,dns.'qry.name == baidu.com'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
DNS:
Topic / Item                  Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
Total Packets                 2                                                       4.2553        100%          0.0200        3.818
 rcode                        2                                                       4.2553        100.00%       0.0200        3.818
  No error                    2                                                       4.2553        100.00%       0.0200        3.818
 opcodes                      2                                                       4.2553        100.00%       0.0200        3.818
  Standard query              2                                                       4.2553        100.00%       0.0200        3.818
 Query/Response               2                                                       4.2553        100.00%       0.0200        3.818
  Response                    1                                                       2.1277        50.00%        0.0100        3.818
  Query                       1                                                       2.1277        50.00%        0.0100        3.818
 Query Type                   2                                                       4.2553        100.00%       0.0200        3.818
  A (Host Address)            2                                                       4.2553        100.00%       0.0200        3.818
 Class                        2                                                       4.2553        100.00%       0.0200        3.818
  IN                          2                                                       4.2553        100.00%       0.0200        3.818
Payload size                  2             60.00         50            70            4.2553        100%          0.0200        3.818
Query Stats                   0                                                       0.0000        100%          -             -
 Qname Len                    1             9.00          9             9             2.1277                      0.0100        3.818
 Label Stats                  0                                                       0.0000                      -             -
  2nd Level                   1                                                       2.1277                      0.0100        3.818
  4th Level or more           0                                                       0.0000                      -             -
  3rd Level                   0                                                       0.0000                      -             -
  1st Level                   0                                                       0.0000                      -             -
Response Stats                0                                                       0.0000        100%          -             -
 no. of questions             2             1.00          1             1             4.2553                      0.0200        3.818
 no. of authorities           2             0.00          0             0             4.2553                      0.0200        3.818
 no. of answers               2             2.00          2             2             4.2553                      0.0200        3.818
 no. of additionals           2             1.00          1             1             4.2553                      0.0200        3.818
Service Stats                 0                                                       0.0000        100%          -             -
 request-response time (msec) 1             0.47          0.470000      0.470000      2.1277                      0.0100        3.818
 no. of unsolicited responses 0                                                       0.0000                      -             -
 no. of retransmissions       0                                                       0.0000                      -             -

--------------------------------------------------------------------------------------------------------------------------------
⊙ 15:53:38 ~/pkgs
```

5）统计分析IP端点（endpoints,ip）

端点将只关注单个数据包中，源目的通信情况。

比如统计IP层，每个IP流入流出流量、包量可以是：

```
tshark -n -q -r <filename> -z endpoints,ip
```

```
 ⊙ 22:39:13  ~/pkgs  tshark -n -q -r retransmission.pcap -z endpoints,ip
Running as user "root" and group "root". This could be dangerous.
================================================================================
IPv4 Endpoints
Filter:<No Filter>
                  |  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.3           205         35774         155          18548          50          17226
114.67.251.123        171         30262         31           14530          140         15732
183.47.125.117         18          2842         10            1938          8            904
101.91.37.19           10          2208         5              441          5           1767
192.168.1.201           4           240         4              240          0              0
224.0.0.18              4           240         0                0          4            240
192.168.1.12            3           264         2              174          1             90
173.194.174.188         2           121         1               66          1             55
192.168.1.99            1            77         1               77          0              0
================================================================================
 ⊙ 22:39:39  ~/pkgs
```

conv则会关注整个会话的情况。

当然，它也能指定过滤规则，比如只统计涉及到TLS传输的数据：

```
tshark -n -q -r <filename> -z endpoints,ip,'tls'
```

```
 ⊙ 22:47:09  ~/pkgs  tshark -n -q -r retransmission.pcap -z endpoints,ip,'tls'
Running as user "root" and group "root". This could be dangerous.
================================================================================
IPv4 Endpoints
Filter:tls
                  |  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.3           148         25134         126          12437          22          12697
114.67.251.123        144         23292         21           12502          123         10790
101.91.37.19            4          1842          1             195          3            1647
================================================================================
 ⊙ 22:47:12  ~/pkgs
```

6）统计分析TCP端点（endpoints,tcp）

将会分析单个数据包的TCP头部维度，汇总统计：

```
tshark -n -q -r <filename> -z endpoints,tcp
```

```
 ⊙ 22:39:39  ~/pkgs  tshark -n -q -r retransmission.pcap -z endpoints,tcp
Running as user "root" and group "root". This could be dangerous.
================================================================================
TCP Endpoints
Filter:<No Filter>
                  |  Port  ||  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.3           58865        171         30262        140          15732          31          14530
114.67.251.123        40616        171         30262        31           14530          140         15732
192.168.1.3           58863         10          2208         5            1767          5             441
101.91.37.19            443         10          2208         5             441          5            1767
192.168.1.12             22          3           264         2             174          1              90
192.168.1.3           57474          3           264         1              90          2             174
192.168.1.3           57219          2           121         1              55          1              66
173.194.174.188        5228          2           121         1              66          1              55
================================================================================
 ⊙ 22:43:10  ~/pkgs
```

既然是TCP传输层，那么显而易见会有端口。

同理，它也支持指定过滤规则，过滤tls或者通信端口为58865、5228的数据可以是：

```
tshark -n -q -r <filename> -z endpoints,tcp,'tls||tcp.port in {58865,5228}'
```

```
 ⊙ 22:48:27  ~/pkgs  tshark -n -q -r retransmission.pcap -z endpoints,tcp,'tls||tcp.port in {58865,5228}'
Running as user "root" and group "root". This could be dangerous.
================================================================================
TCP Endpoints
Filter:tls||tcp.port in {58865,5228}
                  |  Port  ||  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.3           58865        171         30262        140          15732          31          14530
114.67.251.123        40616        171         30262        31           14530          140         15732
192.168.1.3           58863          4          1842         3            1647          1             195
101.91.37.19            443          4          1842         1             195          3            1647
192.168.1.3           57219          2           121         1              55          1              66
173.194.174.188        5228          2           121         1              66          1              55
================================================================================
 ⊙ 22:49:22  ~/pkgs
```

7）统计分析UDP端点（endpoints,udp）

很好，你已经会举一反三了，统计分析UDP端点数据：

```
tshark -n -q -r <filename> -z endpoints,udp
```

```
 ⊙ 22:49:22  ~/pkgs  tshark -n -q -r dns.pcap -z endpoints,udp
Running as user "root" and group "root". This could be dangerous.
================================================================================
UDP Endpoints
Filter:<No Filter>
                  |  Port  ||  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.73            53          6          764          3          469          3          295
255.255.255.255       5001          2          487          0            0          2          487
192.168.1.12         43567          2          306          1          102          1          204
192.168.1.12         36114          2          216          1           98          1          118
192.168.1.12         47969          2          242          1           95          1          147
192.168.1.15         54821          1          305          1          305          0            0
192.168.1.200        52821          1          182          1          182          0            0
192.168.1.11          5678          1          172          1          172          0            0
255.255.255.255       5678          1          172          0            0          1          172
================================================================================
 ⊙ 22:50:43  ~/pkgs
```

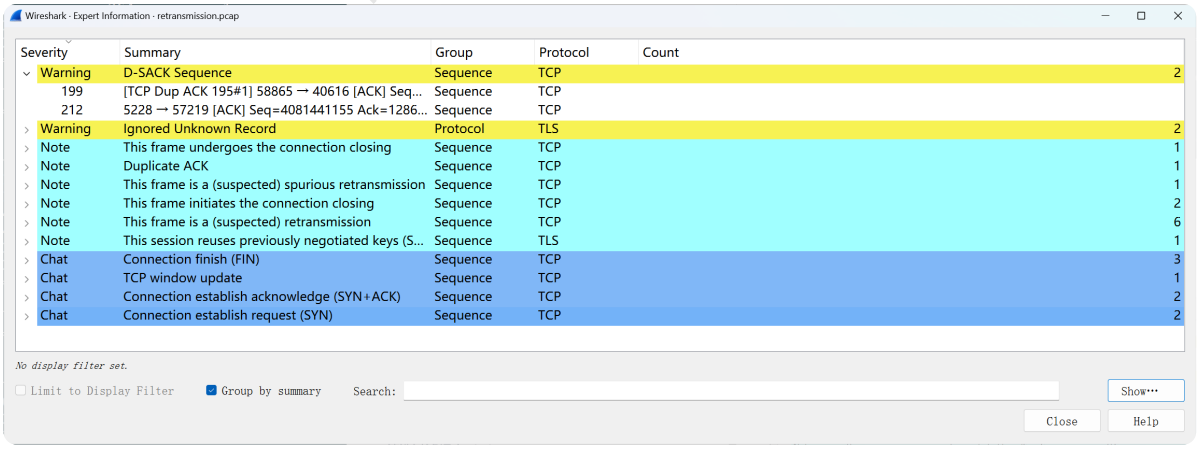
指定过滤规则，只过滤涉及到DNS的数据：

```
tshark -n -q -r <filename> -z endpoints,udp,dns
```

```
 ⊙ 22:50:43  ~/pkgs  tshark -n -q -r dns.pcap -z endpoints,udp,dns
Running as user "root" and group "root". This could be dangerous.
================================================================================
UDP Endpoints
Filter:dns
                  |  Port  ||  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx Bytes |
192.168.1.73            53          6          764          3          469          3          295
192.168.1.12         43567          2          306          1          102          1          204
192.168.1.12         36114          2          216          1           98          1          118
192.168.1.12         47969          2          242          1           95          1          147
================================================================================
 ⊙ 22:54:04  ~/pkgs
```

8）以专家模式输出汇总（expert）

对应wireshark的专家信息功能，在wireshark上的展示如下：



在tshark上的实现则是：

```
tshark -n -q -r <filename> -z expert
```

```
 ⊙ 22:57:10  ~/pkgs  tshark -n -q -r retransmission.pcap -z expert
Running as user "root" and group "root". This could be dangerous.

Warns (2)
=============
   Frequency      Group          Protocol  Summary
           2   Sequence               TCP  D-SACK Sequence

Notes (12)
=============
   Frequency      Group          Protocol  Summary
           1   Sequence               TLS  This session reuses previously negotiated keys (Session resumption)
           6   Sequence               TCP  This frame is a (suspected) retransmission
           2   Sequence               TCP  This frame initiates the connection closing
           1   Sequence               TCP  This frame is a (suspected) spurious retransmission
           1   Sequence               TCP  Duplicate ACK (#1)
           1   Sequence               TCP  This frame undergoes the connection closing

Chats (8)
=============
   Frequency      Group          Protocol  Summary
           1   Sequence               TCP  Connection establish request (SYN): server port 443
           1   Sequence               TCP  Connection establish acknowledge (SYN+ACK): server port 443
           1   Sequence               TCP  Connection establish request (SYN): server port 40616
           1   Sequence               TCP  Connection establish acknowledge (SYN+ACK): server port 40616
           1   Sequence               TCP  TCP window update
           3   Sequence               TCP  Connection finish (FIN)
 ⊙ 22:57:19  ~/pkgs
```

它也支持指定过滤选项，比如只分析第三个流，可以是：

```
tshark -n -q -r <filename> -z expert,'tcp.stream==2'
```

```
 ⊙ 23:02:10  ~/pkgs  tshark -n -q -r retransmission.pcap -z expert,'tcp.stream==2'
Running as user "root" and group "root". This could be dangerous.

Warns (1)
=============
   Frequency      Group          Protocol  Summary
           1   Sequence               TCP  D-SACK Sequence

Notes (11)
=============
   Frequency      Group          Protocol  Summary
           6   Sequence               TCP  This frame is a (suspected) retransmission
           2   Sequence               TCP  This frame initiates the connection closing
           1   Sequence               TCP  This frame is a (suspected) spurious retransmission
           1   Sequence               TCP  Duplicate ACK (#1)
           1   Sequence               TCP  This frame undergoes the connection closing

Chats (6)
=============
   Frequency      Group          Protocol  Summary
           1   Sequence               TCP  Connection establish request (SYN): server port 40616
           1   Sequence               TCP  Connection establish acknowledge (SYN+ACK): server port 40616
           1   Sequence               TCP  TCP window update
           3   Sequence               TCP  Connection finish (FIN)
 ⊙ 23:03:04  ~/pkgs
```

> **注意**：如果不是一条完整的流（比如缺失TCP三次握手），那么则可能不会有任何输出。

过滤警告级别的统计数据，可以是：

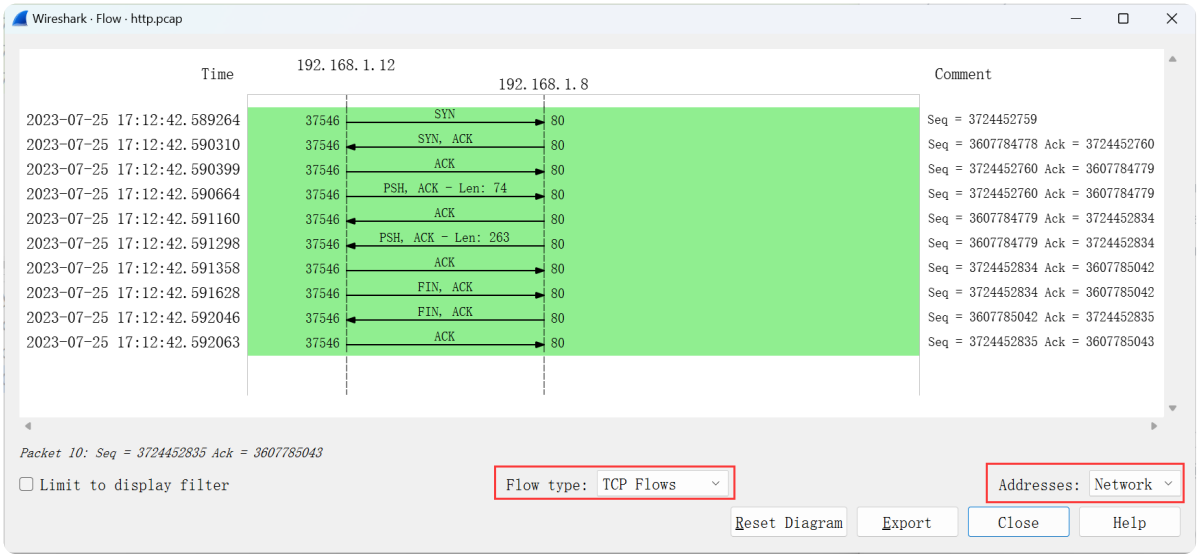
```
tshark -n -q -r <filename> -z expert,'Warns'
```

```
 ⊙ 23:04:18  ~/pkgs  tshark -n -q -r retransmission.pcap -z expert,'Warns'
Running as user "root" and group "root". This could be dangerous.

Warns (2)
=============
   Frequency      Group          Protocol  Summary
           2   Sequence               TCP  D-SACK Sequence
 ⊙ 23:04:43  ~/pkgs
```

9）以流量图形式显示两个端点的通信过程（flow）

对应wireshark的**Flow Graph**功能，即流量图显示功能，可以把整个通信过程画出一个通信图出来，在wireshark上的显示如下：



那么在tshark的实现，可以是：

```
tshark -q -n -r <filename> -z flow,tcp,network
```

```
 23:35:47  ~/pkgs  tshark -q -n -r http.pcap -z flow,tcp,network
Running as user "root" and group "root". This could be dangerous.
|Time      | 192.168.1.12                |                             |
|          |                             | 192.168.1.8                 |
|0.000000  |          SYN                |                             |Seq = 0
|          |(37546)   ------------------>  (80)                       |
|0.001046  |          SYN, ACK           |                             |Seq = 0 Ack = 1
|          |(37546)   <------------------  (80)                       |
|0.001135  |          ACK                |                             |Seq = 1 Ack = 1
|          |(37546)   ------------------>  (80)                       |
|0.001400  |          PSH, ACK - Len: 74 |                             |Seq = 1 Ack = 1
|          |(37546)   ------------------>  (80)                       |
|0.001896  |          ACK                |                             |Seq = 1 Ack = 75
|          |(37546)   <------------------  (80)                       |
|0.002034  |          PSH, ACK - Len: 263|                             |Seq = 1 Ack = 75
|          |(37546)   <------------------  (80)                       |
|0.002094  |          ACK                |                             |Seq = 75 Ack = 264
|          |(37546)   ------------------>  (80)                       |
|0.002364  |          FIN, ACK           |                             |Seq = 75 Ack = 264
|          |(37546)   ------------------>  (80)                       |
|0.002782  |          FIN, ACK           |                             |Seq = 264 Ack = 76
|          |(37546)   <------------------  (80)                       |
|0.002799  |          ACK                |                             |Seq = 76 Ack = 265
|          |(37546)   ------------------>  (80)                       |
 23:35:57  ~/pkgs
```

可以看到，tshark用ASCII字符制表输出，并非真正意义上的图形化界面。

时间上，wireshark默认是**ad**形式，如果你想的话，当然可以通过**-t ad**来指定。

流图形的参数格式为：**flow,name,mode[,filter]**

flow是固定字段，**name**取值范围为：

- any：所有报文
- icmp：icmp

- icmpv6：icmpv6
- lbm_uim：UIM
- tcp：TCP

mode取值范围为：

- standard：所有地址
- network：网络地址

[,filter]则为具体的过滤规则，符合wireshark语法就行。

再来看一个示例，图形化显示ICMP的交互过程：

```
tshark -q -n -r <filename> -z flow,icmp,network
```

```
 ⊙ 23:44:41  ~/pkgs  tshark -q -n -r icmp.pcap -z flow,icmp,network
Running as user "root" and group "root". This could be dangerous.
|Time     | 192.168.1.12                      |
|         |                  | 192.168.1.8    |
|0.000000 |         Echo (ping) request       |ICMP: Echo (ping) request  id=0xb048, seq=3/768, ttl=64
|         |(0)      ----------------->  (2048)|
|0.000089 |         Echo (ping) reply         |ICMP: Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
|         |(0)      <-----------------  (0)   |
|1.001442 |         Echo (ping) request       |ICMP: Echo (ping) request  id=0xb048, seq=4/1024, ttl=64
|         |(0)      ----------------->  (2048)|
|1.001514 |         Echo (ping) reply         |ICMP: Echo (ping) reply    id=0xb048, seq=4/1024, ttl=64 (request in 6)
|         |(0)      <-----------------  (0)   |
 ⊙ 23:44:47  ~/pkgs
```

可以看到一共有两个ICMP request对应两条reply，但报文关联信息字段，只有reply报文的尾部显示了回应给哪个request，**request in 2**表示第二帧，**request in 6**表示第六帧。

结合前面讲到过的二次分析（two-pass）场景，不妨指定**-2**参数看看效果：

```
tshark -q -2 -n -r <filename> -z flow,icmp,network
```

```
 ⊙ 23:52:21  ~/pkgs  tshark -q -2 -n -r icmp.pcap -z flow,icmp,network
Running as user "root" and group "root". This could be dangerous.
|Time     | 192.168.1.12                      |
|         |                  | 192.168.1.8    |
|0.000000 |         Echo (ping) request       |ICMP: Echo (ping) request  id=0xb048, seq=3/768, ttl=64 (reply in 4)
|         |(0)      ----------------->  (2048)|
|0.000089 |         Echo (ping) reply         |ICMP: Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
|         |(0)      <-----------------  (0)   |
|1.001442 |         Echo (ping) request       |ICMP: Echo (ping) request  id=0xb048, seq=4/1024, ttl=64 (reply in 8)
|         |(0)      ----------------->  (2048)|
|1.001514 |         Echo (ping) reply         |ICMP: Echo (ping) reply    id=0xb048, seq=4/1024, ttl=64 (request in 6)
|         |(0)      <-----------------  (0)   |
 ⊙ 23:52:23  ~/pkgs
```

可见request对应的reply在第几个frame，会被完整的填充。

当然，再最后面可以指定过滤规则，比如过滤**icmp.seq==3**：

```
tshark -q -2 -n -r <filename> -z flow,icmp,network,icmp.seq==3
```

```
 ⊙ 23:54:24  ~/pkgs  tshark -q -2 -n -r icmp.pcap -z flow,icmp,network,icmp.seq==3
Running as user "root" and group "root". This could be dangerous.
|Time     | 192.168.1.12                      |
|         |                  | 192.168.1.8    |
|0.000000 |         Echo (ping) request       |ICMP: Echo (ping) request  id=0xb048, seq=3/768, ttl=64 (reply in 4)
|         |(0)      ----------------->  (2048)|
|0.000089 |         Echo (ping) reply         |ICMP: Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
|         |(0)      <-----------------  (0)   |
 ⊙ 23:55:23  ~/pkgs
```

不要通过-Y来指定过滤规则，并不会生效：

```
 ⚙ 23:55:23  ~/pkgs  tshark -q -2 -Y 'icmp.seq==3' -n -r icmp.pcap -z flow,icmp,network
Running as user "root" and group "root". This could be dangerous.
|Time     | 192.168.1.12                  |
|         |                 | 192.168.1.8 |
|0.000000 |        Echo (ping) request    |ICMP: Echo (ping) request  id=0xb048, seq=3/768, ttl=64 (reply in 4)
|         |(0)     ----------------->  (2048)   |
|0.000089 |        Echo (ping) reply      |ICMP: Echo (ping) reply    id=0xb048, seq=3/768, ttl=64 (request in 2)
|         |(0)     <-----------------  (0)      |
|1.001442 |        Echo (ping) request    |ICMP: Echo (ping) request  id=0xb048, seq=4/1024, ttl=64 (reply in 8)
|         |(0)     ----------------->  (2048)   |
|1.001514 |        Echo (ping) reply      |ICMP: Echo (ping) reply    id=0xb048, seq=4/1024, ttl=64 (request in 6)
|         |(0)     <-----------------  (0)      |
 ⚙ 23:56:23  ~/pkgs
```

10）显示两个节点之间的TCP/UDP流内容（follow）

此选项格式为：**follow,prot,mode,filter[,range]**

follow是固定字段，**prot**表示传输协议（Protocols），可取范围如下：

取值	含义
tcp	TCP协议
udp	UDP协议
dccp	DCCP协议
tls	TLS 或 SSL 协议
http	HTTP流
http2	http2流
quic	quic流

mode来指定输出格式，取值如下：

取值	含义
ascii	ASCII格式输出，不可打印字符用点表示
ebcdic	EBCDIC格式输出，不可打印字符用点表示
hex	带偏移量的十六进制和ASCII数据
raw	十六进制数据
yaml	YAML格式

filter用来指定要显示的流，有如下三种格式：

格式	含义
ip-addr0:port0,ip-addr1:port1	指定IP地址和TCP、UDP或DCCP端口对（TCP端口用于TLS、HTTP和HTTP2；QUIC不支持地址和端口匹配）；
stream-index	指定流索引，用于TCP、UDP、DCCP、TLS和HTTP（TLS和HTTP使用TCP流索引）；
stream-index,substream-index	指定流和子流，用于HTTP/2和QUIC，因为它们使用多路复用/长连接（HTTP/2的TCP流和HTTP/2流索引、QUIC连接号和流ID）。

[,range]用于指定应该显示流的哪些部分。

①示例一：以“十六进制”格式显示第一个TCP流的内容

```
tshark -q -n -r <filename> -z "follow,tcp,hex,0"
```

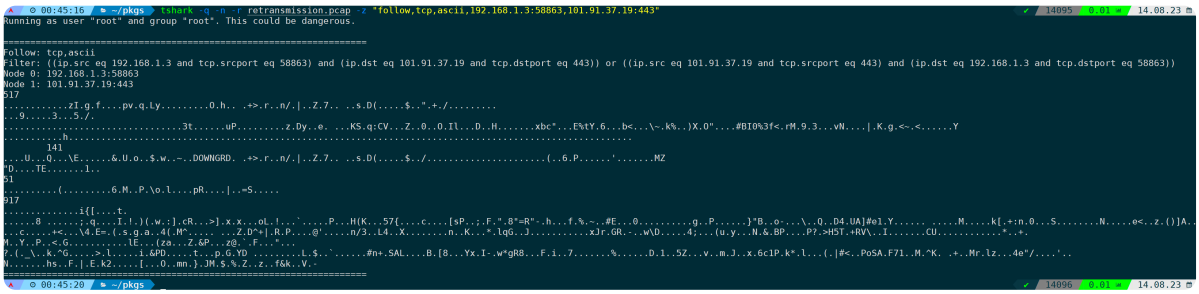
```
 ⊙ 00:35:22  ~/pkgs  tshark -q -n -r http.pcap -z "follow,tcp,hex,0"
Running as user "root" and group "root". This could be dangerous.

===================================================================
Follow: tcp,hex
Filter: tcp.stream eq 0
Node 0: 192.168.1.12:37546
Node 1: 192.168.1.8:80
00000000  47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a  GET / HT TP/1.1..
00000010  48 6f 73 74 3a 20 77 65  62 2d 73 65 72 76 65 72  Host: we b-server
00000020  31 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 63  1..User- Agent: c
00000030  75 72 6c 2f 38 2e 32 2e  30 0d 0a 41 63 63 65 70  url/8.2. 0..Accep
00000040  74 3a 20 2a 2f 2a 0d 0a  0d 0a                    t: */*.. ..
        00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d  HTTP/1.1  200 OK.
        00000010  0a 53 65 72 76 65 72 3a  20 6e 67 69 6e 78 2f 31  .Server:  nginx/1
        00000020  2e 32 33 2e 33 0d 0a 44  61 74 65 3a 20 54 75 65  .23.3..D ate: Tue
        00000030  2c 20 32 35 20 4a 75 6c  20 32 30 32 33 20 30 39  , 25 Jul  2023 09
        00000040  3a 31 32 3a 34 32 20 47  4d 54 0d 0a 43 6f 6e 74  :12:42 G MT..Cont
        00000050  65 6e 74 2d 54 79 70 65  3a 20 74 65 78 74 2f 68  ent-Type : text/h
        00000060  74 6d 6c 0d 0a 43 6f 6e  74 65 6e 74 2d 4c 65 6e  tml..Con tent-Len
        00000070  67 74 68 3a 20 35 0d 0a  4c 61 73 74 2d 4d 6f 64  gth: 5.. Last-Mod
        00000080  69 66 69 65 64 3a 20 46  72 69 2c 20 31 33 20 4a  ified: F ri, 13 J
        00000090  61 6e 20 32 30 32 33 20  30 39 3a 35 36 3a 33 35  an 2023  09:56:35
        000000A0  20 47 4d 54 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   GMT..Co nnection
        000000B0  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 4b 65  : keep-a live..Ke
        000000C0  65 70 2d 41 6c 69 76 65  3a 20 74 69 6d 65 6f 75  ep-Alive : timeou
        000000D0  74 3d 32 30 0d 0a 45 54  61 67 3a 20 22 36 33 63  t=20..ET ag: "63c
        000000E0  31 32 61 64 33 2d 35 22  0d 0a 41 63 63 65 70 74  12ad3-5" ..Accept
        000000F0  2d 52 61 6e 67 65 73 3a  20 62 79 74 65 73 0d 0a  -Ranges:  bytes..
        00000100  0d 0a 74 65 73 74 0a                           ..test.
===================================================================
 ⊙ 00:35:32  ~/pkgs
```

可以清晰看到，**Node0**请求**Node1**的80端口，**Node0**发送了HTTP GET请求给**Node1**，之后拿到了**Node1**的200 OK HTTP响应状态码，上下两段通过多个空格隔开，方便区分。

②示例二：显示IP1:PORT1和IP2:PORT2之间的TCP流的内容

```
tshark -q -n -r <filename> -z "follow,tcp,ascii,IP1:PORT1,IP2:PORT2"
```



这条流为tls流，数据被加密了，所以看不出有意义的内容。从截图不难看出，Filter规则会根据我们写的规则进一步转换填充为wireshark的完整过滤语法，即：

```
((ip.src eq 192.168.1.3 and tcp.srcport eq 58863) and (ip.dst eq 101.91.37.19 and tcp.dstport eq 443)) or ((ip.src eq 101.91.37.19 and tcp.srcport eq 443) and (ip.dst eq 192.168.1.3 and tcp.dstport eq 58863))
```

③示例三：在第一个TCP连接中显示第一个HTTP的内容

```
tshark -q -n -r <filename> -z "follow,http,hex,0,1"
```

```
 01:23:22  ~/pkgs  tshark -q -n -r http.pcap -z "follow,http,hex,0,1"
Running as user "root" and group "root". This could be dangerous.

===================================================================
Follow: http,hex
Filter: tcp.stream eq 0
Node 0: 192.168.1.12:37546
Node 1: 192.168.1.8:80
00000000  47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a  GET / HT TP/1.1..
00000010  48 6f 73 74 3a 20 77 65  62 2d 73 65 72 76 65 72  Host: we b-server
00000020  31 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 63  1..User- Agent: c
00000030  75 72 6c 2f 38 2e 32 2e  30 0d 0a 41 63 63 65 70  url/8.2. 0..Accep
00000040  74 3a 20 2a 2f 2a 0d 0a  0d 0a                    t: */*.. ..
===================================================================
 01:23:42  ~/pkgs
```

第一个HTTP为GET，那么第二个HTTP为response：

```
tshark -q -n -r <filename> -z "follow,http,hex,0,2"
```

```
 01:25:35  ~/pkgs  tshark -q -n -r http.pcap -z "follow,http,hex,0,2"
Running as user "root" and group "root". This could be dangerous.

===================================================================
Follow: http,hex
Filter: tcp.stream eq 0
Node 0: 192.168.1.12:37546
Node 1: 192.168.1.8:80
        00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d  HTTP/1.1  200 OK.
        00000010  0a 53 65 72 76 65 72 3a  20 6e 67 69 6e 78 2f 31  .Server:  nginx/1
        00000020  2e 32 33 2e 33 0d 0a 44  61 74 65 3a 20 54 75 65  .23.3..D ate: Tue
        00000030  2c 20 32 35 20 4a 75 6c  20 32 30 32 33 20 30 39  , 25 Jul  2023 09
        00000040  3a 31 32 3a 34 32 20 47  4d 54 0d 0a 43 6f 6e 74  :12:42 G MT..Cont
        00000050  65 6e 74 2d 54 79 70 65  3a 20 74 65 78 74 2f 68  ent-Type : text/h
        00000060  74 6d 6c 0d 0a 43 6f 6e  74 65 6e 74 2d 4c 65 6e  tml..Con tent-Len
        00000070  67 74 68 3a 20 35 0d 0a  4c 61 73 74 2d 4d 6f 64  gth: 5.. Last-Mod
        00000080  69 66 69 65 64 3a 20 46  72 69 2c 20 31 33 20 4a  ified: F ri, 13 J
        00000090  61 6e 20 32 30 32 33 20  30 39 3a 35 36 3a 33 35  an 2023  09:56:35
        000000A0  20 47 4d 54 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e   GMT..Co nnection
        000000B0  3a 20 6b 65 65 70 2d 61  6c 69 76 65 0d 0a 4b 65  : keep-a live..Ke
        000000C0  65 70 2d 41 6c 69 76 65  3a 20 74 69 6d 65 6f 75  ep-Alive : timeou
        000000D0  74 3d 32 30 0d 0a 45 54  61 67 3a 20 22 36 33 63  t=20..ET ag: "63c
        000000E0  31 32 61 64 33 2d 35 22  0d 0a 41 63 63 65 70 74  12ad3-5" ..Accept
        000000F0  2d 52 61 6e 67 65 73 3a  20 62 79 74 65 73 0d 0a  -Ranges:  bytes..
        00000100  0d 0a                                           ..
===================================================================
 01:25:40  ~/pkgs
```

如果是长连接，一个TCP连接中有多个HTTP请求，修改最后面的数字即可：

```
 14:04:05  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z 'follow,http,hex,0,4'
Running as user "root" and group "root". This could be dangerous.

===================================================================
Follow: http,hex
Filter: tcp.stream eq 0
Node 0: 192.168.1.3:37334
Node 1: 192.168.1.72:8080
000001ED  47 45 54 20 2f 63 73 73  2f 6c 6f 67 69 6e 2e 63  GET /css /login.c
000001FD  73 73 3f 76 3d 31 77 6d  72 6d 33 6e 20 48 54 54  ss?v=1wm rm3n HTT
0000020D  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 31 39 32  P/1.1..H ost: 192
0000021D  2e 31 36 38 2e 31 2e 37  32 3a 38 30 38 30 0d 0a  .168.1.7 2:8080..
0000022D  43 6f 6e 6e 65 63 74 69  6f 6e 3a 20 6b 65 65 70  Connecti on: keep
0000023D  2d 61 6c 69 76 65 0d 0a  53 65 63 2d 50 75 72 70  -alive.. Sec-Purp
0000024D  6f 73 65 3a 20 70 72 65  66 65 74 63 68 3b 70 72  ose: pre fetch;pr
0000025D  65 72 65 6e 64 65 72 0d  0a 55 73 65 72 2d 41 67  erender. .User-Ag
0000026D  65 6e 74 3a 20 4d 6f 7a  69 6c 6c 61 2f 35 2e 30  ent: Moz illa/5.0
0000027D  20 28 57 69 6e 64 6f 77  73 20 4e 54 20 31 30 2e   (Window s NT 10.
0000028D  30 3b 20 57 69 6e 36 34  3b 20 78 36 34 29 20 41  0; Win64 ; x64) A
0000029D  70 70 6c 65 57 65 62 4b  69 74 2f 35 33 37 2e 33  ppleWebK it/537.3
000002AD  36 20 28 4b 48 54 4d 4c  2c 20 6c 69 6b 65 20 47  6 (KHTML , like G
000002BD  65 63 6b 6f 29 20 43 68  72 6f 6d 65 2f 31 31 35  ecko) Ch rome/115
000002CD  2e 30 2e 30 2e 30 20 53  61 66 61 72 69 2f 35 33  .0.0.0 S afari/53
000002DD  37 2e 33 36 0d 0a 41 63  63 65 70 74 3a 20 74 65  7.36..Ac cept: te
000002ED  78 74 2f 63 73 73 2c 2a  2f 2a 3b 71 3d 30 2e 31  xt/css,* /*;q=0.1
000002FD  0d 0a 50 75 72 70 6f 73  65 3a 20 70 72 65 66 65  ..Purpos e: prefe
0000030D  74 63 68 0d 0a 52 65 66  65 72 65 72 3a 20 68 74  tch..Ref erer: ht
0000031D  74 70 3a 2f 2f 31 39 32  2e 31 36 38 2e 31 2e 37  tp://192 .168.1.7
0000032D  32 3a 38 30 38 30 2f 0d  0a 41 63 63 65 70 74 2d  2:8080/. .Accept-
0000033D  45 6e 63 6f 64 69 6e 67  3a 20 67 7a 69 70 2c 20  Encoding : gzip,
0000034D  64 65 66 6c 61 74 65 0d  0a 41 63 63 65 70 74 2d  deflate. .Accept-
0000035D  4c 61 6e 67 75 61 67 65  3a 20 7a 68 2d 43 4e 2c  Language : zh-CN,
0000036D  7a 68 3b 71 3d 30 2e 39  2c 65 6e 3b 71 3d 30 2e  zh;q=0.9 ,en;q=0.
0000037D  38 0d 0a 0d 0a                                    8....
===================================================================
 14:04:20  ~/pkgs
```

> 此时还是在第一条TCP流中(tcp.stream==0)，过滤的第4条HTTP请求的内容。

11）统计分析HTTP状态（http,stat）

统计分析HTTP的状态码以及请求方法，可以使用如下命令：

```
tshark -q -n -r <filename> -z http,stat
```

```
 14:04:20  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http,stat
Running as user "root" and group "root". This could be dangerous.

================================================================
HTTP Statistics
* HTTP Response Status Codes              Packets
  200 OK                                      133
  404 Not Found                                 1
* HTTP Request Methods                    Packets
  GET                                         133
  POST                                          1
================================================================
 14:06:28  ~/pkgs
```

可以看到整个报文中，有133个200 OK、1个404状态码，以及133个GET和1个POST请求方法。

它自然也支持应用过滤规则，比如只过滤第一条TCP流对应的HTTP状态码和请求方法，可以是：

```
tshark -q -n -r <filename> -z http,stat,'tcp.stream==0'
```

```
 14:06:28  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http,stat,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================================================
HTTP Statistics with filter tcp.stream==0
* HTTP Response Status Codes                 Packets
  200 OK                                          19
* HTTP Request Methods                       Packets
  GET                                             19
================================================================================
 14:09:13  ~/pkgs
```

12）统计分析HTTP树状结构（http,tree）

统计HTTP数据包分布。输出结果为响应状态代码和请求方法：

```
tshark -q -n -r <filename> -z http,tree
```

```
 14:45:26  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http,tree
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       Burst Rate    Burst Start
------------------------------------------------------------------------------------------------------------------------------------------------
Total HTTP Packets     268                                                     0.0055        100%          1.4800        7.636
 HTTP Response Packets 134                                                     0.0028        50.00%        0.7400        7.639
  2xx: Success         133                                                     0.0027        99.25%        0.7300        7.639
   200 OK              133                                                     0.0027        100.00%       0.7300        7.639
  4xx: Client Error    1                                                       0.0000        0.75%         0.0100        7.726
   404 Not Found       1                                                       0.0000        100.00%       0.0100        7.726
  ???: broken          0                                                       0.0000        0.00%         -             -
  5xx: Server Error    0                                                       0.0000        0.00%         -             -
  3xx: Redirection     0                                                       0.0000        0.00%         -             -
  1xx: Informational   0                                                       0.0000        0.00%         -             -
 HTTP Request Packets  134                                                     0.0028        50.00%        0.7400        7.636
  GET                  133                                                     0.0027        99.25%        0.7400        7.636
  POST                 1                                                       0.0000        0.75%         0.0100        7.483
 Other HTTP Packets    0                                                       0.0000        0.00%         -             -

------------------------------------------------------------------------------------------------------------------------------------------------
 14:47:05  ~/pkgs
```

当然也支持在最后面加过滤规则。如果是HTTP2协议，则使用**http2,tree**。

13）统计分析HTTP请求的URI路径（http_req,tree）

http_req,tree 只会统计请求涉及到的URI资源路径，不关注响应：

```
tshark -q -n -r <filename> -z http_req,tree
```

```
 14:47:05  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_req,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================================
HTTP/Requests:
Topic / Item                                                Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
------------------------------------------------------------------------------------------------------------------------------------------------
HTTP Requests by HTTP Host                                  134                                                     0.0028        100%          0.7400        7.636
 192.168.1.72:8080                                          133                                                     0.0027        99.25%        0.7400        7.636
  /                                                         2                                                       0.0000        1.50%         0.0100        0.000
  /views/transferlist.html                                  1                                                       0.0000        0.75%         0.0100        7.707
  /views/search.html                                        1                                                       0.0000        0.75%         0.0100        7.718
  /views/rss.html                                           1                                                       0.0000        0.75%         0.0100        7.710
  /views/propertiesToolbar.html                             1                                                       0.0000        0.75%         0.0100        7.741
  /views/properties.html                                    1                                                       0.0000        0.75%         0.0100        7.741
  /views/filters.html                                       1                                                       0.0000        0.75%         0.0100        7.703
  /scripts/prop-webseeds.js                                 1                                                       0.0000        0.75%         0.0100        7.818
  /scripts/prop-trackers.js                                 1                                                       0.0000        0.75%         0.0100        7.818
  /scripts/prop-peers.js                                    1                                                       0.0000        0.75%         0.0100        7.818
  /scripts/prop-general.js                                  1                                                       0.0000        0.75%         0.0100        7.818
  /scripts/prop-files.js                                    1                                                       0.0000        0.75%         0.0100        7.818
  /scripts/progressbar.js?v=1wmrm3n                         1                                                       0.0000        0.75%         0.0100        7.631
  /scripts/preferences.js                                   1                                                       0.0000        0.75%         0.0100        7.624
  /scripts/mocha-init.js?locale=zh&v=1wmrm3n                1                                                       0.0000        0.75%         0.0100        7.631
  /scripts/misc.js?locale=zh&v=1wmrm3n                      1                                                       0.0000        0.75%         0.0100        7.631
  /scripts/login.js?locale=zh&v=1wmrm3n                     1                                                       0.0000        0.75%         0.0100        0.000
  /scripts/lib/mocha.min.js                                 1                                                       0.0000        0.75%         0.0100        7.624
  /scripts/lib/clipboard.min.js                             1                                                       0.0000        0.75%         0.0100        7.631
  /scripts/lib/MooTools-More-1.6.0-compat-compressed.js     1                                                       0.0000        0.75%         0.0100        7.624
  /scripts/lib/MooTools-Core-1.6.0-compat-compressed.js     1                                                       0.0000        0.75%         0.0100        7.612
  /scripts/filesystem.js?v=1wmrm3n                          1                                                       0.0000        0.75%         0.0100        7.631
```

指定规律规则，只过滤第一条TCP连接的数据：

```
tshark -q -n -r <filename> -z http_req,tree,'tcp.stream==0'
```

```
⊙ 14:52:39  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_req,tree,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================================
HTTP/Requests:
Topic / Item                              Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
------------------------------------------------------------------------------------------------------------------------------------------------
HTTP Requests by HTTP Host                19                                                    0.0024        100%          0.1300        7.655
 192.168.1.72:8080                        19                                                    0.0024        100.00%       0.1300        7.655
  /views/propertiesToolbar.html           1                                                     0.0001        5.26%         0.0100        7.741
  /scripts/prop-general.js                1                                                     0.0001        5.26%         0.0100        7.818
  /scripts/lib/mocha.min.js               1                                                     0.0001        5.26%         0.0100        7.624
  /scripts/file-tree.js?v=1wmrm3n         1                                                     0.0001        5.26%         0.0100        7.631
  /images/handle-icon-horizontal.gif      1                                                     0.0001        5.26%         0.0100        7.719
  /images/bg-handle-horizontal.gif        1                                                     0.0001        5.26%         0.0100        7.708
  /images/bg-dropdown.gif                 1                                                     0.0001        5.26%         0.0100        7.666
  /images/arrow-right.gif                 1                                                     0.0001        5.26%         0.0100        7.734
  /icons/media-playback-pause.svg         1                                                     0.0001        5.26%         0.0100        7.655
  /icons/kt-magnet.svg                    1                                                     0.0001        5.26%         0.0100        7.659
  /icons/go-bottom.svg                    1                                                     0.0001        5.26%         0.0100        7.657
  /icons/filterstalled.svg                1                                                     0.0001        5.26%         0.0100        7.732
  /icons/edit-delete.svg                  1                                                     0.0001        5.26%         0.0100        7.665
  /icons/edit-copy.svg                    1                                                     0.0001        5.26%         0.0100        7.662
  /icons/downloading.svg                  1                                                     0.0001        5.26%         0.0100        7.728
  /icons/dialog-warning.svg               1                                                     0.0001        5.26%         0.0100        7.745
  /css/style.css?v=1wmrm3n                1                                                     0.0001        5.26%         0.0100        7.610
  /css/login.css?v=1wmrm3n                1                                                     0.0001        5.26%         0.0100        0.000
  /                                       1                                                     0.0001        5.26%         0.0100        0.000

------------------------------------------------------------------------------------------------------------------------------------------------
⊙ 14:54:26  ~/pkgs
```

14）统计分析HTTP请求URL（http_seq,tree）

和**http_req,tree**的区别是，它会自动补齐请求的服务器，以URL路径方式呈现出来：

```
tshark -q -n -r <filename> -z http_seq,tree
```

```
⊙ 14:56:58  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_seq,tree                                                                    ✔ 14180
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================================
HTTP/Request Sequences:
Topic / Item                                                                  Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
------------------------------------------------------------------------------------------------------------------------------------------------
HTTP Request Sequences                                                        132                                                   0.0027        100%          0.7400        7.636
 http://192.168.1.72:8080/                                                    137                                                   0.0028        103.79%       0.7400        7.636
  http://192.168.1.72:8080/                                                   245                                                   0.0050        178.83%       1.3800        7.636
   http://192.168.1.72:8080/css/Layout.css?v=1wmrm3n                          13                                                    0.0003        5.31%         0.1200        7.666
    http://192.168.1.72:8080/images/handle-icon.gif                           1                                                     0.0000        7.69%         0.0100        7.700
    http://192.168.1.72:8080/images/handle-icon-horizontal.gif                1                                                     0.0000        7.69%         0.0100        7.719
    http://192.168.1.72:8080/images/bg-panel-header.gif                       1                                                     0.0000        7.69%         0.0100        7.707
    http://192.168.1.72:8080/images/bg-handle-horizontal.gif                  1                                                     0.0000        7.69%         0.0100        7.708
    http://192.168.1.72:8080/images/bg-dropdown.gif                           1                                                     0.0000        7.69%         0.0100        7.666
    http://192.168.1.72:8080/icons/collapse.svg                               1                                                     0.0000        7.69%         0.0100        7.708
   http://192.168.1.72:8080/css/style.css?v=1wmrm3n                           7                                                     0.0001        2.86%         0.0600        7.666
    http://192.168.1.72:8080/images/toolbox-divider.gif                       1                                                     0.0000        14.29%        0.0100        7.666
    http://192.168.1.72:8080/images/arrow-right.gif                           1                                                     0.0000        14.29%        0.0100        7.734
    http://192.168.1.72:8080/icons/edit-find.svg                              1                                                     0.0000        14.29%        0.0100        7.667
   http://192.168.1.72:8080/css/Tabs.css?v=1wmrm3n                            3                                                     0.0001        1.22%         0.0300        7.612
    http://192.168.1.72:8080/images/tabs.gif                                  1                                                     0.0000        33.33%        0.0100        7.667
   http://192.168.1.72:8080/views/transferlist.html                           1                                                     0.0000        0.41%         0.0100        7.707
   http://192.168.1.72:8080/views/search.html                                 1                                                     0.0000        0.41%         0.0100        7.718
   http://192.168.1.72:8080/views/rss.html                                    1                                                     0.0000        0.41%         0.0100        7.710
   http://192.168.1.72:8080/views/propertiesToolbar.html                      1                                                     0.0000        0.41%         0.0100        7.741
   http://192.168.1.72:8080/views/properties.html                             1                                                     0.0000        0.41%         0.0100        7.741
   http://192.168.1.72:8080/views/filters.html                                1                                                     0.0000        0.41%         0.0100        7.703
```

同样支持指定过滤规则：

```
tshark -q -n -r <filename> -z http_seq,tree,'tcp.stream==0'
```

```
⊙ 15:01:24  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_seq,tree,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================================
HTTP/Request Sequences:
Topic / Item                                                                  Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
------------------------------------------------------------------------------------------------------------------------------------------------
HTTP Request Sequences                                                        18                                                    0.0023        100%          0.1300        7.655
 http://192.168.1.72:8080/                                                    29                                                    0.0037        161.11%       0.1900        7.655
  http://192.168.1.72:8080/css/style.css?v=1wmrm3n                            3                                                     0.0004        10.34%        0.0200        7.734
   http://192.168.1.72:8080/images/arrow-right.gif                            1                                                     0.0001        33.33%        0.0100        7.734
  http://192.168.1.72:8080/views/propertiesToolbar.html                       1                                                     0.0001        3.45%         0.0100        7.741
  http://192.168.1.72:8080/scripts/prop-general.js                            1                                                     0.0001        3.45%         0.0100        7.818
  http://192.168.1.72:8080/scripts/lib/mocha.min.js                           1                                                     0.0001        3.45%         0.0100        7.624
  http://192.168.1.72:8080/scripts/file-tree.js?v=1wmrm3n                     1                                                     0.0001        3.45%         0.0100        7.631
  http://192.168.1.72:8080/icons/media-playback-pause.svg                     1                                                     0.0001        3.45%         0.0100        7.655
  http://192.168.1.72:8080/icons/kt-magnet.svg                                1                                                     0.0001        3.45%         0.0100        7.659
  http://192.168.1.72:8080/icons/go-bottom.svg                                1                                                     0.0001        3.45%         0.0100        7.657
  http://192.168.1.72:8080/icons/filterstalled.svg                            1                                                     0.0001        3.45%         0.0100        7.732
  http://192.168.1.72:8080/icons/edit-delete.svg                              1                                                     0.0001        3.45%         0.0100        7.665
  http://192.168.1.72:8080/icons/edit-copy.svg                                1                                                     0.0001        3.45%         0.0100        7.662
  http://192.168.1.72:8080/icons/downloading.svg                              1                                                     0.0001        3.45%         0.0100        7.728
  http://192.168.1.72:8080/icons/dialog-warning.svg                           1                                                     0.0001        3.45%         0.0100        7.745
  http://192.168.1.72:8080/css/login.css?v=1wmrm3n                            1                                                     0.0001        3.45%         0.0100        0.000
 http://192.168.1.72:8080/css/Layout.css?v=1wmrm3n                            6                                                     0.0008        33.33%        0.0600        7.666
  http://192.168.1.72:8080/images/handle-icon-horizontal.gif                  1                                                     0.0001        16.67%        0.0100        7.719
  http://192.168.1.72:8080/images/bg-handle-horizontal.gif                    1                                                     0.0001        16.67%        0.0100        7.708
  http://192.168.1.72:8080/images/bg-dropdown.gif                             1                                                     0.0001        16.67%        0.0100        7.666

------------------------------------------------------------------------------------------------------------------------------------------------
⊙ 15:02:18  ~/pkgs
```

15）统计计算HTTP请求和响应（http_srv,tree）

对于HTTP request，显示的值是目的端服务器IP地址和服务器主机名。对于HTTP response，显示的值是目的端服务器IP地址及状态：

```
tshark -q -n -r <filename> -z http_srv,tree
```

```
 ⊙ 15:03:10  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_srv,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
HTTP/Load Distribution:
Topic / Item                    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
HTTP Requests by Server         134                                                     0.0028        100%          0.7400        7.636
 HTTP Requests by Server Address 134                                                    0.0028        100.00%       0.7400        7.636
  192.168.1.72                  133                                                     0.0027        99.25%        0.7400        7.636
   192.168.1.72:8080            133                                                     0.0027        100.00%       0.7400        7.636
  192.168.1.81                  1                                                       0.0000        0.75%         0.0100        40.192
   blog.linux-code.com          1                                                       0.0000        100.00%       0.0100        40.192
 HTTP Requests by HTTP Host     134                                                     0.0028        100.00%       0.7400        7.636
  192.168.1.72:8080             133                                                     0.0027        99.25%        0.7400        7.636
   192.168.1.72                 133                                                     0.0027        100.00%       0.7400        7.636
  blog.linux-code.com           1                                                       0.0000        0.75%         0.0100        40.192
   192.168.1.81                 1                                                       0.0000        100.00%       0.0100        40.192
HTTP Responses by Server Address 134                                                    0.0028        100%          0.7400        7.639
 192.168.1.72                   133                                                     0.0027        99.25%        0.7400        7.639
  OK                            132                                                     0.0027        99.25%        0.7300        7.639
  KO                            1                                                       0.0000        0.75%         0.0100        7.726
 192.168.1.81                   1                                                       0.0000        0.75%         0.0100        40.561
  OK                            1                                                       0.0000        100.00%       0.0100        40.561

--------------------------------------------------------------------------------------------------------------------------------
 ⊙ 15:03:15  ~/pkgs
```

也支持在最后指定过滤规则，统计过滤第一条流：

```
tshark -q -n -r <filename> -z http_srv,tree,'tcp.stream==0'
```

```
 ⊙ 15:08:38  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z http_srv,tree,tcp.stream==0
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
HTTP/Load Distribution:
Topic / Item                    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
HTTP Requests by Server         19                                                      0.0024        100%          0.1300        7.655
 HTTP Requests by Server Address 19                                                     0.0024        100.00%       0.1300        7.655
  192.168.1.72                  19                                                      0.0024        100.00%       0.1300        7.655
   192.168.1.72:8080            19                                                      0.0024        100.00%       0.1300        7.655
 HTTP Requests by HTTP Host     19                                                      0.0024        100.00%       0.1300        7.655
  192.168.1.72:8080             19                                                      0.0024        100.00%       0.1300        7.655
   192.168.1.72                 19                                                      0.0024        100.00%       0.1300        7.655
HTTP Responses by Server Address 19                                                     0.0024        100%          0.1300        7.655
 192.168.1.72                   19                                                      0.0024        100.00%       0.1300        7.655
  OK                            19                                                      0.0024        100.00%       0.1300        7.655

--------------------------------------------------------------------------------------------------------------------------------
 ⊙ 15:08:40  ~/pkgs
```

16）统计分析ICMP（icmp,srt）

计算ICMP回显请求总数、回复、丢失和丢失百分比，以及ping返回的最小、最大、平均、中值和样本标准差SRT统计信息。

```
tshark -q -n -r <filename> -z icmp,srt
```

```
 ⊙ 15:11:56  ~/pkgs  tshark -q -n -r icmp.pcap -z icmp,srt
Running as user "root" and group "root". This could be dangerous.

================================================================================
ICMP Service Response Time (SRT) Statistics (all times in ms):
Filter: <none>

Requests  Replies   Lost      % Loss
24        21        3          12.5%

Minimum   Maximum   Mean      Median    SDeviation     Min Frame Max Frame
0.416     0.945     0.657     0.625     0.134          21        11
================================================================================
 ⊙ 15:12:27  ~/pkgs
```

过滤某个IP地址的统计：

```
tshark -q -n -r <filename> -z icmp,srt,'ip.addr==xxx'
```

```
 15:12:27  ~/pkgs  tshark -q -n -r icmp.pcap -z icmp,srt,'ip.addr==192.168.1.106'
Running as user "root" and group "root". This could be dangerous.

================================================================================
ICMP Service Response Time (SRT) Statistics (all times in ms):
Filter: ip.addr==192.168.1.106

Requests  Replies   Lost      % Loss
3         0         3         100.0%

Minimum   Maximum   Mean      Median    SDeviation     Min Frame Max Frame
0.000     0.000     0.000     0.000     0.000          0         0
================================================================================
 15:16:55  ~/pkgs  tshark -q -n -r icmp.pcap -z icmp,srt,'ip.addr==192.168.1.1'
Running as user "root" and group "root". This could be dangerous.

================================================================================
ICMP Service Response Time (SRT) Statistics (all times in ms):
Filter: ip.addr==192.168.1.1

Requests  Replies   Lost      % Loss
7         7         0           0.0%

Minimum   Maximum   Mean      Median    SDeviation     Min Frame Max Frame
0.484     0.945     0.666     0.662     0.158          8         11
================================================================================
 15:17:00  ~/pkgs
```

如果是V6地址，则使用**icmpv6,srt**。

17）统计协议层次结构及包量（io,phs）

创建协议层次结构统计信息，列出数据包数和字节数：

```
tshark -q -n -r <filename> -z io,phs
```

```
 15:17:00  ~/pkgs  tshark -q -n -r icmp.pcap -z io,phs
Running as user "root" and group "root". This could be dangerous.

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

sll                                      frames:48 bytes:4704
  ip                                     frames:48 bytes:4704
    icmp                                 frames:48 bytes:4704
================================================================
 15:19:41  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,phs
Running as user "root" and group "root". This could be dangerous.

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

eth                                      frames:554 bytes:379233
  ip                                     frames:554 bytes:379233
    tcp                                  frames:554 bytes:379233
      http                               frames:268 bytes:183747
        data-text-lines                  frames:21 bytes:17662
          tcp.segments                   frames:9 bytes:5491
        media                            frames:19 bytes:12673
          tcp.segments                   frames:19 bytes:12673
        xml                              frames:60 bytes:63717
          tcp.segments                   frames:2 bytes:548
        urlencoded-form                  frames:1 bytes:539
        image-gif                        frames:8 bytes:5488
          tcp.segments                   frames:1 bytes:1146
        json                             frames:26 bytes:17777
          tcp.segments                   frames:1 bytes:1429
================================================================
 15:19:50  ~/pkgs
```

只统计第一条TCP流：

```
tshark -q -n -r <filename> -z io,phs,'tcp.stream==0'
```

```
 15:19:50  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,phs,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================================
Protocol Hierarchy Statistics
Filter: tcp.stream==0

eth                                      frames:75 bytes:56223
  ip                                     frames:75 bytes:56223
    tcp                                  frames:75 bytes:56223
      http                               frames:38 bytes:26431
        data-text-lines                  frames:4 bytes:3605
          tcp.segments                   frames:1 bytes:226
        media                            frames:3 bytes:2488
          tcp.segments                   frames:3 bytes:2488
        xml                              frames:8 bytes:8565
        image-gif                        frames:4 bytes:2386
================================================================
 15:23:30  ~/pkgs
```

18）统计分析包量和字节大小（io,stat）

格式为：**io,stat,interval[,filter]**

其中**io,stat**是固定的，interval表示间隔时间，可以指定秒或小数秒或微秒，如果指定为0，将计算所有数据包的统计信息。

统计所有数据包的包量和字节大小：

```
tshark -q -n -r <filename> -z io,stat,0
```

```
 15:31:08  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,stat,0
Running as user "root" and group "root". This could be dangerous.

===================================
| IO Statistics                   |
|                                 |
| Duration: 56.7 secs             |
| Interval: 56.7 secs             |
|                                 |
| Col 1: Frames and bytes         |
|---------------------------------|
|              |1                 |
| Interval     | Frames |  Bytes |
|---------------------------------|
|  0.0 <> 56.7 |    554 | 379233 |
===================================
 15:31:10  ~/pkgs  tshark -q -n -r icmp.pcap -z io,stat,0
Running as user "root" and group "root". This could be dangerous.

===================================
| IO Statistics                   |
|                                 |
| Duration: 41.7 secs             |
| Interval: 41.7 secs             |
|                                 |
| Col 1: Frames and bytes         |
|---------------------------------|
|              |1               | |
| Interval     | Frames | Bytes | |
|------------------------------| | |
|  0.0 <> 41.7 |     48 |  4704 | |
===================================
 15:31:47  ~/pkgs
```

如果想指定间隔为10s统计一次，且只统计第一条TCP流则可以是：

```
tshark -q -n -r <filename> -z io,stat,10,'tcp.stream==0'
```

```
 15:35:13  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,stat,10,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

============================
| IO Statistics            |
|                          |
| Duration: 56.699790 secs |
| Interval: 10 secs        |
|                          |
| Col 1: tcp.stream==0     |
|--------------------------|
|          |1              |
| Interval | Frames | Bytes |
|--------------------------|
|  0 <> 10 |     71 | 55995 |
| 10 <> 20 |      4 |   228 |
| 20 <> 30 |      0 |     0 |
| 30 <> 40 |      0 |     0 |
| 40 <> 50 |      0 |     0 |
| 50 <> Dur|      0 |     0 |
============================
 15:35:19  ~/pkgs
```

19）统计分析某个字段的最大最小平均值等（MAX|MIN|AVG）

依然是io,stat，但完整格式为：**io,stat,interval,"COUNT|SUM|MIN|MAX|AVG|LOAD(field)filter"**

io,stat是固定选项，**interval**表示间隔时间，其中的COUNT|SUM|MIN|MAX|AVG|LOAD都要基于过滤规则来做统计，比如：

```
io,stat,0,'MAX(icmp.data_time_relative)icmp.data_time_relative'
```

统计的是icmp响应时间字段，并取最大值MAX，0表示统计所有，不分间隔。

而如果要指定某个过滤规则，在最后面加，比如只统计到某个IP的最大的ICMP响应值情况，完整的命令为：

```
tshark -q -n -r <filename> -z
io,stat,0,'MAX(icmp.data_time_relative)icmp.data_time_relative,'ip.addr==xxx''
```

```
⊙ 15:52:14 ~/pkgs tshark -q -n -r icmp.pcap -z io,stat,0,'MAX(icmp.data_time_relative)icmp.data_time_relative,'ip.addr==192.168.1.1''
Running as user "root" and group "root". This could be dangerous.

============================================================
| IO Statistics                                            |
|                                                          |
| Duration: 41.7 secs                                      |
| Interval: 41.7 secs                                      |
|                                                          |
| Col 1: MAX(icmp.data_time_relative)icmp.data_time_relative |
|     2: ip.addr==192.168.1.1                              |
|----------------------------------------------------------|
|                |1        |2                |             |
| Interval       |    MAX  | Frames | Bytes  |             |
|----------------------------------------|                 |
|  0.0 <> 41.7 | 0.694421 |     17 |  1704 |             |
============================================================
⊙ 15:52:18 ~/pkgs
```

可以看到最大值为0.69s，注意单位是秒。

又或者，统计第一条TCP连接中，距离它上一个包间隔时间最长的为：

```
tshark -q -n -r <filename> -z
io,stat,0,'MAX(tcp.time_delta)tcp.time_delta,tcp.stream==0'
```

```
⊙ 15:53:35 ~/pkgs tshark -q -n -r http-keep-alive.pcap -z io,stat,0,'MAX(tcp.time_delta)tcp.time_delta,tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

==========================================
| IO Statistics                          |
|                                        |
| Duration: 56.7 secs                    |
| Interval: 56.7 secs                    |
|                                        |
| Col 1: MAX(tcp.time_delta)tcp.time_delta |
|     2: tcp.stream==0                   |
|----------------------------------------|
|              |1        |2              |
| Interval     |    MAX  | Frames | Bytes |
|----------------------------------------|
|  0.0 <> 56.7 | 8.914911 |     75 | 56223 |
==========================================
⊙ 15:56:31 ~/pkgs
```

统计第一条TCP流中，HTTP响应时间的平均值和最大值，分别为：

```
tshark -q -n -r <filename> -z io,stat,0,'AVG(http.time)http.time,tcp.stream==0'
```

```
tshark -q -n -r <filename> -z io,stat,0,'MAX(http.time)http.time,tcp.stream==0'
```

```
 ⊙ 15:58:37  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,stat,0,'AVG(http.time)http.time,tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================
| IO Statistics                                |
|                                              |
| Duration: 56.7 secs                          |
| Interval: 56.7 secs                          |
|                                              |
| Col 1: AVG(http.time)http.time               |
|     2: tcp.stream==0                         |
|----------------------------------------------|
|                |1          |2                |
| Interval       |    AVG    | Frames | Bytes  |
|----------------------------------------------|
|  0.0 <> 56.7   | 0.007198  |     75 | 56223  |
================================================
 ⊙ 15:58:45  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,stat,0,'MAX(http.time)http.time,tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================
| IO Statistics                                |
|                                              |
| Duration: 56.7 secs                          |
| Interval: 56.7 secs                          |
|                                              |
| Col 1: MAX(http.time)http.time               |
|     2: tcp.stream==0                         |
|----------------------------------------------|
|                |1          |2                |
| Interval       |    MAX    | Frames | Bytes  |
|----------------------------------------------|
|  0.0 <> 56.7   | 0.369137  |     75 | 56223  |
================================================
 ⊙ 15:58:52  ~/pkgs
```

如果不指定过滤规则（上图是tcp.stream==0）则统计所有包含http.time字段的报文。

再或者，只统计返回了200 OK状态码的http最大响应时间：

```
tshark -q -n -r <filename> -z io,stat,0,'MAX(http.time)http.time and
http.response.code == 200'
```

```
 ⊙ 16:05:32  ~/pkgs  tshark -q -n -r http-keep-alive.pcap -z io,stat,0,'MAX(http.time)http.time and http.response.code == 200'
Running as user "root" and group "root". This could be dangerous.

============================================================
| IO Statistics                                            |
|                                                          |
| Duration: 56.7 secs                                      |
| Interval: 56.7 secs                                      |
|                                                          |
| Col 1: MAX(http.time)http.time and http.response.code == 200 |
|----------------------------------------------------------|
|              |1          |                               |
| Interval     |    MAX    |                               |
|--------------|           |                               |
|  0.0 <> 56.7 | 0.369137  |                               |
============================================================
 ⊙ 16:05:56  ~/pkgs
```

20）统计IP地址占比（ip_hosts,tree）

源和目的IP地址都会在这里展示，并显示百分比、发包速率、开始时间等：

```
tshark -q -n -r <filename> -z ip_hosts,tree
```

```
⊙ 16:05:56 ~/pkgs tshark -q -n -r http-keep-alive.pcap -z ip_hosts,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/All Addresses:
Topic / Item   Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
All Addresses  554                                                    0.0098        100%          2.8200        7.610
 192.168.1.3   554                                                    0.0098        100.00%       2.8200        7.610
 192.168.1.72  532                                                    0.0094        96.03%        2.8200        7.610
 192.168.1.81  22                                                     0.0004        3.97%         0.1400        40.560

--------------------------------------------------------------------------------------------------------------------------------
⊙ 16:10:29 ~/pkgs tshark -q -n -r icmp.pcap -z ip_hosts,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/All Addresses:
Topic / Item    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
All Addresses   48                                                     0.0012        100%          0.0300        41.674
 192.168.1.12   48                                                     0.0012        100.00%       0.0300        41.674
 192.168.1.1    17                                                     0.0004        35.42%        0.0200        8.169
 192.168.1.73   8                                                      0.0002        16.67%        0.0200        30.222
 192.168.1.11   8                                                      0.0002        16.67%        0.0200        15.966
 192.168.1.81   6                                                      0.0001        12.50%        0.0200        26.339
 192.168.1.106  3                                                      0.0001        6.25%         0.0100        34.755
 192.168.1.82   2                                                      0.0000        4.17%         0.0200        41.735
 192.168.1.6    2                                                      0.0000        4.17%         0.0200        0.000
 192.168.1.200  2                                                      0.0000        4.17%         0.0200        20.939

--------------------------------------------------------------------------------------------------------------------------------
⊙ 16:13:09 ~/pkgs
```

在最后指定过滤规则，比如只统计icmp序列号小于等于6的报文和只统计第一条TCP流的报文，分别是：

```
tshark -q -n -r <filename> -z ip_hosts,tree,'icmp.seq <= 6'
```

```
tshark -q -n -r <filename> -z ip_hosts,tree,'tcp.stream==0'
```

```
⊙ 16:16:03 ~/pkgs tshark -q -n -r icmp.pcap -z ip_hosts,tree,'icmp.seq <= 6'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/All Addresses:
Topic / Item    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
All Addresses   35                                                     0.0012        100%          0.0200        8.169
 192.168.1.12   35                                                     0.0012        100.00%       0.0200        8.169
 192.168.1.1    12                                                     0.0004        34.29%        0.0200        8.169
 192.168.1.11   8                                                      0.0003        22.86%        0.0200        15.966
 192.168.1.81   6                                                      0.0002        17.14%        0.0200        26.339
 192.168.1.73   6                                                      0.0002        17.14%        0.0200        30.325
 192.168.1.106  3                                                      0.0001        8.57%         0.0100        34.755

--------------------------------------------------------------------------------------------------------------------------------
⊙ 16:16:34 ~/pkgs tshark -q -n -r http-keep-alive.pcap -z ip_hosts,tree,'tcp.stream==0'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/All Addresses:
Topic / Item   Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
All Addresses  75                                                     0.0044        100%          0.4200        7.610
 192.168.1.72  75                                                     0.0044        100.00%       0.4200        7.610
 192.168.1.3   75                                                     0.0044        100.00%       0.4200        7.610

--------------------------------------------------------------------------------------------------------------------------------
⊙ 16:17:08 ~/pkgs
```

如果是IPv6地址，则使用 **ipv6_hosts,tree**。

21）统计源地址和目标地址占比（ip_srcdst,tree）

将源地址和目的地址区分开来，则使用**ip_srcdst,tree**：

```
tshark -q -n -r <filename> -z ip_srcdst,tree
```

```
⊙ 19:59:31 ~/pkgs tshark -q -n -r http-keep-alive.pcap -z ip_srcdst,tree
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/Source and Destination Addresses:
Topic / Item                    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
Source IPv4 Addresses           554                                                     0.0098        100%          2.8200        7.610
 192.168.1.72                   298                                                     0.0053        53.79%        1.8400        7.589
 192.168.1.3                    241                                                     0.0043        43.50%        1.0100        7.610
 192.168.1.81                   15                                                      0.0003        2.71%         0.1200        40.560
Destination IPv4 Addresses      554                                                     0.0098        100%          2.8200        7.610
 192.168.1.3                    313                                                     0.0055        56.50%        1.8400        7.589
 192.168.1.72                   234                                                     0.0041        42.24%        1.0100        7.610
 192.168.1.81                   7                                                       0.0001        1.26%         0.0300        40.191

--------------------------------------------------------------------------------------------------------------------------------
⊙ 20:00:40 ~/pkgs
```

指定过滤规则，只过滤tcp端口为8080的数据：

```
tshark -q -n -r <filename> -z ip_srcdst,tree,'tcp.port in {8080}'
```

```
⊙ 20:02:33 ~/pkgs tshark -q -n -r http-keep-alive.pcap -z ip_srcdst,tree,'tcp.port in {8080}'
Running as user "root" and group "root". This could be dangerous.

================================================================================================================================
IPv4 Statistics/Source and Destination Addresses:
Topic / Item                    Count         Average       Min Val       Max Val       Rate (ms)     Percent       Burst Rate    Burst Start
--------------------------------------------------------------------------------------------------------------------------------
Source IPv4 Addresses           532                                                     0.0094        100%          2.8200        7.610
 192.168.1.72                   298                                                     0.0053        56.02%        1.8400        7.589
 192.168.1.3                    234                                                     0.0041        43.98%        1.0100        7.610
Destination IPv4 Addresses      532                                                     0.0094        100%          2.8200        7.610
 192.168.1.3                    298                                                     0.0053        56.02%        1.8400        7.589
 192.168.1.72                   234                                                     0.0041        43.98%        1.0100        7.610

--------------------------------------------------------------------------------------------------------------------------------
⊙ 20:02:37 ~/pkgs
```

如果是IPv6地址，则使用 **ip6_srcdst,tree**。

11.完整展开报文（-V）

-V参数会完整展开报文的每个字段、头部信息.

比如下面这个示例，将第一次握手的请求展开：

```
tshark -n -r <filename> -V -Y 'tcp.flags.syn==1&&tcp.flags.ack==0&&frame.number<=5'
```

```
 ⊙ 20:19:37  ~/pkgs  tshark -n -r http-keep-alive.pcap -V -Y 'tcp.flags.syn==1&&tcp.flags.ack==0&&frame.number<=5'
Running as user "root" and group "root". This could be dangerous.
Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface unknown, id 0
    Section number: 1
    Interface id: 0 (unknown)
        Interface name: unknown
    Encapsulation type: Ethernet (1)
    Arrival Time: Aug 14, 2023 13:58:37.582083000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1691992717.582083000 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: 66 bytes (528 bits)
    Capture Length: 66 bytes (528 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
Ethernet II, Src: 50:eb:f6:3c:36:14, Dst: 00:50:56:81:be:58
    Destination: 00:50:56:81:be:58
        Address: 00:50:56:81:be:58
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 50:eb:f6:3c:36:14
        Address: 50:eb:f6:3c:36:14
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.72
    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 (0)
    Total Length: 52
    Identification: 0x5cb0 (23728)
    010. .... = Flags: 0x2, 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: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.3
    Destination Address: 192.168.1.72
```

```
Transmission Control Protocol, Src Port: 37334, Dst Port: 8080, Seq: 0, Len: 0
    Source Port: 37334
    Destination Port: 8080
    [Stream index: 0]
    [Conversation completeness: Incomplete (0)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 469605482
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x002 (SYN)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Accurate ECN: Not set
        .... 0... .... = Congestion Window Reduced: Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...0 .... = Acknowledgment: Not set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..1. = Syn: Set
            [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 8080]
                [Connection establish request (SYN): server port 8080]
                [Severity level: Chat]
                [Group: Sequence]
        .... .... ...0 = Fin: Not set
        [TCP Flags: ··········S·]
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x83c2 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
        TCP Option - Maximum segment size: 1460 bytes
            Kind: Maximum Segment Size (2)
            Length: 4
            MSS Value: 1460
        TCP Option - No-Operation (NOP)
            Kind: No-Operation (1)
        TCP Option - Window scale: 8 (multiply by 256)
            Kind: Window Scale (3)
            Length: 3
            Shift count: 8
            [Multiplier: 256]
        TCP Option - No-Operation (NOP)
            Kind: No-Operation (1)
        TCP Option - No-Operation (NOP)
            Kind: No-Operation (1)
        TCP Option - SACK permitted
            Kind: SACK Permitted (4)
            Length: 2
    [Timestamps]
        [Time since first frame in this TCP stream: 0.000000000 seconds]
        [Time since previous frame in this TCP stream: 0.000000000 seconds]
 ⊙ 20:19:43  ~/pkgs
```

所有单个报文都能被展开，被当初数据批量过滤处理，那么就能配合Linux下的文本过滤命令，来批量拿到我们想要的字段信息。

比如通过正则来拿到我们想要的直播流URL：

```
tshark -n -r <filename> -Y 'http.request.method eq GET' -V | grep -Po '(?<=Full
request URI:\s).*m3u8(?=])' |& sort -u
```

```
 ⊙ 20:26:15  ~/pkgs  tshark -n -r iptv.pcap -Y 'http.request.method eq GET' -V | grep -Po '(?<=Full request URI:\s).*m3u8(?=])' |& sort -u
Running as user "root" and group "root". This could be dangerous.
http://101.207.178.17:80/PLTV/88888888/224/3221227373/1.m3u8
http://101.207.178.18:80/PLTV/88888888/224/3221227479/1.m3u8
http://101.207.178.27:80/PLTV/88888888/224/3221227477/1.m3u8
http://101.207.178.2:80/PLTV/88888888/224/3221227435/1.m3u8
http://101.207.178.2:80/PLTV/88888888/224/3221227479/1.m3u8
http://sc.rrs.169ol.com:80/PLTV/88888888/224/3221227373/index.m3u8
http://sc.rrs.169ol.com:80/PLTV/88888888/224/3221227435/index.m3u8
http://sc.rrs.169ol.com:80/PLTV/88888888/224/3221227477/index.m3u8
http://sc.rrs.169ol.com:80/PLTV/88888888/224/3221227479/index.m3u8
 ⊙ 20:26:46  ~/pkgs
```

报文展开后，所有字段都能被当成普通文本处理，因此非常方便，这是图形化wireshark所做不到的。

12.保存报文（-w）

顾名思义，通过tshark处理完原始报文后，需要将过滤出来的报文保存到新的抓包文件，那么此参数就派上用场了。

比如过滤前300个报文，并且IP地址为A或B，或者目的端口为38388的HTTP报文，可以是：

```
tshark -n -r <filename> -Y 'frame.number<=300&&(ip.addr in
{101.207.176.11,192.168.137.111}||tcp.dstport==38388)&&http'
```

```
 ⊙ 20:41:04  ~/pkgs  tshark -n -r iptv.pcap -Y 'frame.number<=300&&(ip.addr in {101.207.176.11,192.168.137.111}||tcp.dstport==38388)&&http'
Running as user "root" and group "root". This could be dangerous.
  154  22.460163 192.168.137.111 → 202.99.114.14 HTTP 492 POST /PORTAL/dsm/loginByUserId HTTP/1.1  (text/plain)
  160  22.505018 202.99.114.14 → 192.168.137.111 HTTP/JSON 59 HTTP/1.1 200  , JavaScript Object Notation (application/json)
  168  22.548980 192.168.137.111 → 101.207.176.11 HTTP 215 GET / HTTP/1.1
  171  22.553018 101.207.176.11 → 192.168.137.111 HTTP 666 HTTP/1.1 200 OK  (text/html)
  180  22.581680 192.168.137.111 → 202.99.114.30 HTTP 294 POST /PORTAL/dsm/heartbeat HTTP/1.1
  183  22.599223 192.168.137.111 → 202.99.114.30 HTTP 409 POST /PORTAL/dsm/getAPKClientVersion HTTP/1.1  (text/plain)
  186  22.608655 192.168.137.111 → 202.99.114.30 HTTP 364 POST /PORTAL/dsm/getAppsByPay HTTP/1.1  (text/plain)
  189  22.610704 202.99.114.30 → 192.168.137.111 HTTP/JSON 59 HTTP/1.1 200  , JavaScript Object Notation (application/json)
  196  22.637780 202.99.114.30 → 192.168.137.111 HTTP/JSON 59 HTTP/1.1 200  , JavaScript Object Notation (application/json)
  198  22.640369 202.99.114.30 → 192.168.137.111 HTTP/JSON 59 HTTP/1.1 200  , JavaScript Object Notation (application/json)
  212  22.746988 192.168.137.111 → 202.99.114.30 HTTP/JSON 416 POST /PORTAL/dsm/getAPKClientVersion HTTP/1.1 , JavaScript Object Notation (application/json)
  225  23.024656 202.99.114.30 → 192.168.137.111 HTTP/JSON 59 HTTP/1.1 200  , JavaScript Object Notation (application/json)
  300  24.753706 192.168.137.111 → 119.6.201.6  HTTP 552 POST /client/upgrade/reportApps? HTTP/1.1  (application/x-www-form-urlencoded)
 ⊙ 20:41:30  ~/pkgs
```

通过-w参数来保存到一个新的抓包文件http_in_300.pcap：

```
tshark -n -r <filename> -Y 'frame.number<=300&&(ip.addr in
{101.207.176.11,192.168.137.111}||tcp.dstport==38388)&&http' -w http_in_300.pcap
```

```
 ⊙ 20:41:30  ~/pkgs  tshark -n -r iptv.pcap -Y 'frame.number<=300&&(ip.addr in {101.207.176.11,192.168.137.111}||tcp.dstport==38388)&&http' -w http_in_300.pcap
Running as user "root" and group "root". This could be dangerous.
 ⊙ 20:44:09  ~/pkgs  ls -lh --full-time http_in_300.pcap
-rw-r--r-- 1 root root 4.4K 2023-08-14 20:44:09.431165536 +0800 http_in_300.pcap
 ⊙ 20:44:29  ~/pkgs  ls -lh --full-time iptv.pcap
-rw-r--r-- 1 root root 516M 2023-07-22 12:50:45.000000000 +0800 iptv.pcap
 ⊙ 20:44:37  ~/pkgs
```

最终保存到的新抓包文件只有4.4K大小，原始文件为516MB。

四、总结

tshark作为wireshark的命令行版本，很多功能其实都是一对一息息相关的，但tshark提供了命令行能力，对于自动化脚本分析有很大的帮助，可以轻松实现自动批量化处理抓包文件，并展示分析结果。

通过过滤、分析网络数据包，它为网络管理员、安全专家和开发人员提供了深入了解网络协议的机会。无论是监控网络流量、故障排除、安全审计还是协议分析，tshark都能发挥其巨大作用。从基本的捕获和过滤到高级的Lua定制，它提供了广泛的功能和应用，适用于各种场景。

本文详细讲述了每个参数的用法及使用示例，篇幅原因无法将每种可能写进去，网络协议本身具备多样性和复杂性。通过掌握tshark其用法，再去分析协议特征，通过对协议的理解和对tshark本身的融会贯通，相信对于各大网络排障都能从中受益。