

Wireshark 命令行工具 tshark (1.12.1) 命令中文详解:

作者: [leolovenet@gmail.com](mailto:leolovenet@gmail.com)  
blog: <http://www.leolovenet.com>  
weibo: <http://weibo.com/leolovenet>  
github: <https://github.com/leolovenet/linux-stuffs>  
版本: 0.1

该命令可以使用两种过滤器。以 `-f` 和 `-R/-Y` 选项后面的内容作为“过滤器表达式”。

`-f` 选项后为“捕获过滤器”，表示在捕获数据包时，只捕获指定的内容。该过滤器遵循 `pcap` 库的捕获过滤器表达式规则，具体的描述可以查看帮助: `man pcap-filter`

`-R/-Y` 选项为“展示过滤器”，表示只将捕获数据中感兴趣的内容展示出来。该过滤器遵循的时 `wireshark` 自己的过滤器表达式，具体的规则可以查看帮助: `man wireshark-filter`

选项:

`-2` 执行 two-pass 分析。

这导致 tshark 缓存输出内容，直到 first pass 完成，这样允许 tshark 可以填充一些需要 future knowledge 才能填充的字段，像 'response in frame #' 字段。这还允许 tshark 通过正确地计算重组帧(reassembly frame)。

该选项不能应用于 Live Capture, 可以应用于读取之前捕获文件的情况。

`-a <capture autostop condition>` 指定 tshark 什么时候自动停止捕获。具体的格式可以下面中的一种:

`duration:value`, 在 `value`秒以后停止捕获。

`filesize:value`, 在捕获的文件到达 `value/kB` 以后停止。如果这个选项跟 `-b` 选项一起使用的话, tshark将在捕获的文件达到上限后, 开启一个新的捕获文件。如果是读取之前捕获的数据包文件时, 当读取的文件大小超过这个限制后, tshark将会停止读取剩余的捕获数据包。 `filesize`的最大值为 2GB。

`files:value`, 在 `value`个捕获文件被写入后, 将会停止捕获。

`-b <capture ring buffer option>` 指定 tshark 将捕获的文件保存为多个。此选项的值跟 `-a`选项的值很相似。具体看一下 man里的说明。

例如: `-b filesize:1000 -b files:5` 指定将捕获的数据存储到5个文件中, 每个文件1M。

`-B <capture buffer size>` 指定捕获缓存大小, 默认为2M。如果在捕获是遇到了包丢失的情况, 可以试着增大这个选项的值。这个选项可以出现多次, 如果它出现在第一个 `-i` 选项之前, 表示设置的为默认“捕获缓存大小”。如果它出现在 `-i` 选项之后, 它设置的为本选项出现之前到最后一个 `-i` 选项指定的网口的“缓存大小”。

`-c(小写) <capture packet count>` 指定最多捕获数据包个数, 如果情况为读取捕获包文件时, 表示最多读取的包个数。

`-C(大写) <configuration profile>` 加载特定的首选项配置文件, 并运行。

`-d <layer type>===<selector>,<decode-as protocol>` 指定怎样“强制解析一个层”数据。类似 `wireshark` 的“Decode as ...”功能。

例如: `-d tcp.port==8888,http` 会将端口 8888 上的数据解析为http协议。

`-d tcp.port==8888:3,http` 会将端口 8888\8889\8890 上的数据解析为http协议。

`-d tcp.port==8888-8890,http` 同上

打印出所有有效的 selector 命令: `tshark -d .`

查看所有默认的 `<layer type>,<selector>,<decode-as protocol>`列表, 可以使用命令:

`tshark -G decodes`

`-D` 打印出所有可捕获数据包的网口列表。

`-e <field>` 如果指定了 `-T fields` 选项(当 `-T` 选项的值为 `fields`时), 可以通过此选项指定那些字段被显示出来。

本选项可以使用多次, 当 `-T fields` 选项被指定后, 那么应该至少使用一次本选项选择要显示的字段。

字段列的名字可能需要添加前缀 “\_ws.col.”

例如:

`-e frame.number -e ip.addr -e udp -e _ws.col.info`

相对于给出单个要打印的字段, 如果给出了一个协议的名字, 那么关于这个协议的多个字段会被当做一个独立的字段被打印出来。

字段之间默认是用 tab 间隔的。可以通过 `-E` 选项控制打印字段的格式。

关于所有可用的候选 `fields` 名可以通过 `tshark -G fields` 命令输出的第三列值获取,例如获取所有http的候选field

`tshark -G fields |grep -w http`

`-E <field print option>` 如果指定了 `-T fields` 选项(当 `-T` 选项的值为 `fields`), 通过设定本选项的值来控制打印出来信息字段的格式。

字段打印选项可以是如下格式:

`header=y|n` 如果指定 `y`, 会在显示结果的第一个行打印出 `-e` 选项指定字段的的名字列表。

这个名字列表将会同样使用本选项指定的分隔符来分隔。

`sparator=/t|/s|<character>` 指定用于分隔字段的分隔符。

如果指定 `/t` 将会使用tab, 这个是默认行为。

如果指定 `/s` 将会使用一个单独的空格。

否则任何可以被命令行接受的字符都可以作为分隔符。

`occurrence=f|l|a` 设定当有多个同样地字段出现时使用哪个。

如果指定 `f` 将会使用第一个出现的字段。

如果指定 `l` 将会使用最后一个出现的字段。

如果指定 `a` 所有出现的字段都将被使用, 这个是默认行为。

`aggregator=,|/s|<character>` 设定当有多个同样地字段出现时使用的“聚合字符 (aggregator)

character) ”。

如果指定 `,` 将会使用逗号, 这个是默认行为。

如果指定 `/s` 将会使用一个单独的空格。

否则任何可以被命令行接受的字符都可以作为分隔符。

`quote=d|s|n` 设定围绕字段的引用字符。

如果指定 `d` 将会使用一对双引号, 包裹字段。

如果指定 `s` 将会使用一对单引号, 包裹字段。

如果指定 `n` 不会使用引用字符, 这是默认行为。

`-f <capture filter>` 指定“捕获过滤器表达式”。该表达式的规则可以查看: `man pcap-filter`

这个选项可以出现多次, 如果它出现在第一个 `-i` 选项之前, 表示设置的为默认“捕获过滤器表达式”。如果它出现在 `-i` 选项之后, 它设置的为本选项出现之前到最后一个 `-i` 选项指定的网口的“捕获过滤器表达式”。

`-F <file format>` 当使用 `-w` 选项将捕获到的数据包, 保存到一个文件时, 同时使用此选项可以指定保存的文件格式。

直接运行 `tshark -F` 将会列出所有支持的文件格式。

`-g` 该选项将保存的捕获文件设为组权限可读。意味着跟你同组的其他用户将有权限读取此文件。当你使用一个非root账号登陆, 并使用 `sudo tshark` 捕获包时, 捕获到的数据文件, 当前用户是没有权限读取的, 这时此选项就很有用了。

`-G [column-formats|currentprefs|decodes|defaultprefs|fields|ftypes|heuristic-decodes|plugins|protocols|values]`

可以将该选项命名为“tshark的信息查询选项”, `tshark`很多功能都有很多候选值可以选择, 而`-G`可以列出有那些候选值可供我们使用。

可用的查询类型:

`column-formats`

列出 `tshark` 可以理解的“列格式”, 你可以用 `-o` 选项配合使用本选项里Field 1 的值, 覆盖默认首选项的值, 然后对`tshark`的输出数据进行格式化。

Field 1 格式化字符串 (例如: `%rD`)

Field 2 对这个格式化字符串的文字描述。(例如: “目的端口 (resolved) ”)

例如: 使 `tshark` 的输出格式为 `wireshark` 的默认输出列格式:

`tshark -o`

```
'gui.column.format:"No.", "%m", "Time", "%t", "Source", "%s", "Destination", "%d", "Protocol", "%p", "Length", "%L", "Info", "%i"'
```

`currentprefs`

列出 `tshark` 当前的首选项配置文件内容。

`decodes`

看 `-d` 选项的解释。

`defaultprefs`

列出 `tshark` 默认首选项配置文件内容。

`fields`

这个选项将会列出很多很长的内容, 为 `tshark` 注册的全部数据库信息, 表示 `tshark` 支持的协议和字段内容。

具体格式, 可以分为两大类: `Protocols(协议) & Header Fields(头字段)`。

如果列出的信息每行的第一个字母为“`P`”表示该记录为一个 `Protocol` 记录。格式为:

Field 1 = '`P`'

Field 2 = descriptive protocol name(协议名描述)

Field 3 = protocol abbreviation(缩写)

例子:

Field 1	Field 2	Field 3
P	Virtual Network Computing	vnc
P	Hypertext Transfer Protocol	http
P	HyperText Transfer Protocol 2	http2

查看所有支持的协议: `tshark -G fields|egrep "^P"`

如果列出的信息每行的第一个字母为“`F`”表示该记录为一个 `Header Fields` 记录。格式为:

Field 1 = '`F`'

Field 2 = descriptive field name(头字段名的描述)

Field 3 = field abbreviation(头字段缩写)

Field 4 = type (头字段的数据类型, 值为`tshark -G ftypes`之一)

Field 5 = parent protocol abbreviation(该字段的父协议的缩写)

Field 6 = 当协议使用一个或多个字节中不同的二进制位表示不同的信息时, Field 6值描述一组相关信息在"parent bitfield width(父协议位字段的宽度)" for `FT_BOOLEAN`。看下面的例子。

Field 7 = bitmask: format: hex: 0x.... 描述Field 3代表的信息在Field 6宽的所有比特位中占用那一位。

Field 8 = blurb describing field(简介信息描述)

例子:

Field1	Field2	Field3	Field4	Field5	Field6	Field7	Field8
F	Mouse button #1 position	vnc.button_1_pos	FT_BOOLEAN	vnc	8	0x1	Whether mouse button #1 is being pressed or not
F	Mouse button #2 position	vnc.button_2_pos	FT_BOOLEAN	vnc	8	0x2	Whether mouse button #2 is being pressed or not
F	Mouse button #3 position	vnc.button_3_pos	FT_BOOLEAN	vnc	8	0x4	Whether mouse button #3 is being pressed or not
F	Mouse button #4 position	vnc.button_4_pos	FT_BOOLEAN	vnc	8	0x8	Whether mouse button #4 is being pressed or not
F	Mouse button #5 position	vnc.button_5_pos	FT_BOOLEAN	vnc	8	0x10	Whether mouse button #5 is being pressed or not
F	Mouse button #6 position	vnc.button_6_pos	FT_BOOLEAN	vnc	8	0x20	Whether mouse button #6 is being pressed or not
F	Mouse button #7 position	vnc.button_7_pos	FT_BOOLEAN	vnc	8	0x40	Whether mouse button #7 is being pressed or not
F	Mouse button #8 position	vnc.button_8_pos	FT_BOOLEAN	vnc	8	0x80	Whether mouse button #8 is being pressed or not

vnc协议使用总 8 个比特位来表示鼠标的8个事件, 0x1 表示8个比特位中只有第一位为1, 0x2 表示只有第二位为1, 0x4表示只有第三位为1.

查看所有支持的字段值: `tshark -G fields|egrep "^F"`

查看ip协议支持的字段: `tshark -G fields |awk -F'\t' '$5 == "ip" {print $0}'`

查看tcp协议支持的字段: `tshark -G fields |awk -F'\t' '$5 == "tcp" {print $0}'`

ftype

列出 tshark 可以理解的基础数据类型。看上面 fields 里 Header Fields 类型的 Field 4 的解释。

heuristic-decodes

列出当前安装的试探解码器信息。每条记录的格式:

- Field 1 = underlying dissector(原始解码器) (e.g. "tcp")
- Field 2 = name of heuristic decoder(试探解码器) (e.g. http)
- Field 3 = heuristic enabled(是否启用试探解码器) (e.g. "True" or "False")

plugins

显示当前安装的插件列表。格式为:

插件库的名字	插件库版本	插件类型	插件库的路径
gryphon.so	0.0.4	dissector	/usr/local/Cellar/wireshark/1.12.1/lib/wireshark/plugins/1.12.1/gryphon.so
stats_tree.so	0.0.1	tap	/usr/local/Cellar/wireshark/1.12.1/lib/wireshark/plugins/1.12.1/stats_tree.so

protocols

显示所有在数据库中注册的协议。格式为:

协议名	协议简写	协议过滤器名
Hypertext Transfer Protocol	HTTP	http
SSH Protocol	SSH	ssh
Domain Name Service	DNS	dns

values

列出一个 fields 里 Header Fields 类型的 Field 3 可能的值。

值得类型有三种: value\_strings(值字符串), range\_strings(范围字符串), true/false strings(布尔字符串)

这三种数据类型分别对应本选项显示结果的三种记录, 记录格式的第一个字段表明本条记录为那种数据类型。

“v” 对应 value\_strings, 格式为:

字段v	fields的缩写(Field 3的值)	可能的整型值	字符串说明
V	vnc.security_type	0	Invalid
V	vnc.security_type	1	None
V	vnc.security_type	2	VNC
V	vnc.security_type	30	Apple Remote Desktop

“R” 对应 range\_strings

字段R	fields的缩写(Field 3的值)	下限整型值	上限整型值	字符串说明
R	mp4ves.start_code	0x0	0x1f	video_object_start_code
R	mp4ves.start_code	0x20	0x2f	video_object_layer_start_code
R	mp4ves.start_code	0x30	0xaf	reserved

R	mp4ves.start_code	0xb0	0xb0	visual_object_sequence_start_code
R	mp4ves.start_code	0xb1	0xb1	visual_object_sequence_end_code
R	mp4ves.start_code	0xb2	0xb2	user_data_start_code
R	mp4ves.start_code	0xb3	0xb3	group_of_vop_start_code
R	mp4ves.start_code	0xb4	0xb4	video_session_error_code
R	mp4ves.start_code	0xb5	0xb5	visual_object_start_code

“T” 对应 true/false

字段T fields的缩写(Field 3的值) True String False String

例如:

T	tcp.flags.res	Set	Not set
T	tcp.flags.ns	Set	Not set
T	tcp.flags.cwr	Set	Not set
T	tcp.flags.ecn	Set	Not set
T	tcp.flags.urg	Set	Not set
T	tcp.flags.ack	Set	Not set
T	tcp.flags.push	Set	Not set
T	tcp.flags.reset	Set	Not set
T	tcp.flags.syn	Set	Not set
T	tcp.flags.fin	Set	Not set

-H <input hosts file> 读取一个hosts文件的记录内容, 这将会写入到捕获的文件中。暗示开启-W n, 可以被调用多次。

-i <capture interface> | - 指定需要捕获数据的网卡接口名或者管道名。

该选项的候选列表为 tshark -D 结果中得一个。如果没有明确指定接口的话, 将会选择第一个非 non-loopback 的接口。管道名, 要不然的是

FIFO (named pipe)名字要不然就是 “-” 代表从标准输入读取。管道读取的数据必须是标准的 pcap 格式。当同时从多个网口捕获数据时, 该选项可以使用多次, 但是捕获保存的文件将会使 pcap-ng 格式。

-I 将无线网卡设置为 “monitor mode”, 这个选项只支持在802.11Wi-Fi接口。

注意: “monitor mode” 状态将会使无线网卡不能访问原来的网络, 如果之前有使用网络的应用 (例如, 网络硬盘, DNS解析等), 但本机又没有其他网口连接到网络, 那你将不能再访问网络, 那些应用也不能再继续工作。

这个选项可以使用多次, 如果出现在第一个 -i 选项之前, 表示设置全部的无线网卡为 “monitor mode”。如果出现在 -i 选项之后, 表示设置是个网口为 “monitor mode”。

-K <keytab> 现从指定的 keytab 文件中加载 kerberos 加密密钥。这个选项可以使用多次, 用来从多个文件中加载密钥。

例如: -K krb5.keytab

-l <keytab> 当每一个包的信息被打印出来以后, 刷新标准输出的缓存。(严格上讲, 如果 -v 选项也给出的话, 此选项并不是 line-buffered, 但是如果 -v 选项没有给出, 每个数据包又只打印一行信息, 它又很像 line-buffered。该选项通常被使用在将捕获到的数据, 使用管道, 导入到其他程序或者脚本上, 这样当包数据被截获并解析后, 会立即输入给这些程序处理, 而不是等到标准输出缓存被填满以后再输出。)

-L 列出该网口支持的 data link types(数据链路类型,例如:RAW(Raw IP),EN10MB(Ethernet)) 然后退出。这些信息可以被 -y 选项使用。

-n 禁止IP或端口的名字解析。-N 选项可以覆盖该选项的配置。

-N <name resolving flags> 只针对特定的地址或者端口号进行名字解析, 关闭对其他地址或端口的名字解析。

如果 -n 和 -N 选项都给出的话, 此选项将覆盖 -n 选项的值。如果都没有给出的话, 默认所有地址和端口都解析。

该选项的值为可能包含下面字母的字符串:

- c to enable concurrent (asynchronous) DNS lookups
- m to enable MAC address resolution
- n to enable network address resolution
- N to enable using external resolvers (e.g., DNS) for network address resolution
- t to enable transport-layer port number resolution

-o <preference>:<value> 设置覆盖默认首选项的值。

这个选项的值为 prefname:value 格式。prefname为首选项中得名字。value 为对应选项的值。

-O <protocols> 类似 -v 选项, 但是相对于 -v 显示所有协议的详细信息, 该选项导致 tshark 只显示指定协议 (Protocols)的详细信息, 协议以缩写形式给出, 可以是多个, 以逗号分隔的列表。

对于所有可以使用的协议名缩写, 可以使用 tshark -G protocols 来查看,

-p(小) 指定不将网口设置为 “混杂模式”。这个选项可以出现多次, 如果出现在第一个 -i 选项之前表示所有的网口都将不进入混杂模式。

-P(大) 指定即使设置了 -w 选项, 将捕获的数据保存到一个文件时, 同时也在屏幕打印捕获到的包信息。

-q 关闭标准输出到屏幕的信息显示。不能跟-P(大) 选项一起使用。

当使用 -w 选项将捕获到的数据保存到一个文件时, 屏幕只显示当先捕获到的数据包个数, 使用该选项可以关闭这个显示。

当捕获一个数据流, 而不保存到文件, 或者读取一个之前保存的捕获文件时, 屏幕会显示一个自增的数字开头的行, 表示捕获到的数据, 使用该选项可以关闭显示。

这很有用, 当你在使用 -z 选项去计算统计信息, 而又不想这些包信息被打印出来干扰你。

如果想要显示的话, 可以按 ctrl + t 键向 tshark 发送 SIGINFO signal 来显示。

-Q 当捕获数据包时, 只显示 true errors。

这导致使用 `-q` 选项还少的信息输出，网口的名字，捕获到的包个数也都被隐藏，不打印了。

`-r <infile>` 从之前捕获到数据包文件 `infile` 中读取数据，可以是任意支持的数据格式（包括 `gzipped` 格式）。这里还可以使用“管道”或标准输入 `stdin(-)`，但此时就不能再使用压缩过的捕获包了。

`-R <Read filter>` 跟 `-f` 选项类似，只不过是指定“读取过滤器表达式”，而不是“捕获过滤器表达式”。该选项只和 `-2` 选项一起工作。如果不使用 `-2` 选项的话，可以用 `-y` 选项来指定“读取过滤器表达式”。注意：当使用这个选项后，数据包的前置字段查询功能(forward-looking fields, 像 `'response in frame #'`) 将不能在使用, 因为当应用这个过滤器的时候，那些功能还没有被执行。

`-s <capture snaplen>` 设定当捕获数据包时的最大长度。大于指定值的数据将会截断，不保存到内存，或者文件。默认值为 `65535`，这样会捕获所有的包内容。设定 `0` 也等于设定 `65535`

`-S <separator>` Set the line separator to be printed between packets.

`-t a|ad|adoy|d|dd|e|r|u|ud|udoy` 设定屏幕显示捕获数据包信息时的时间戳格式。格式可以是：

- `a absolute`: 你所在时区的绝对时间，没有日期信息。是数据包被捕获的真正时间。  
例如: `21:14:08.333551`
- `ad absolute with date`: 你所在时区的绝对时间，并显示日期信息。格式为: `YYYY-MM-DD Time`。是数据包被捕获的真正时间。  
例如: `2014-10-27 21:14:27.460535`
- `adoy absolute with date using day of year`: 你所在时区的绝对时间，格式为 `YYYY/DOY Time`。是数据包被捕获的真正时间。  
例如: `2014/300 21:23:04.052135`
- `d delta`: 自从上个数据包被捕获的延迟时间。  
例如: `0.000296`
- `dd delta_displayed`: 自从上个展示的数据包被捕获的延迟时间。  
例如: `0.000089`
- `e epoch`: 自从 (Jan 1, 1970 00:00:00) 到现在的秒数。  
例如: `1414417101.327984`
- `r relative`: 从第一个被捕获的数据包到现在的相对时间。  
例如: `1.190222`
- `u UTC`: 数据包被捕获时UTC格式的绝对时间。没有日期信息。  
例如: `13:43:47.697318`
- `ud UTC with date`: 数据包被捕获时UTC格式的绝对时间。并伴随日期信息。格式为: `YYYY-MM-DD Time`  
例如: `2014-10-27 13:46:01.022098`
- `udoy UTC with date using day of year`:  
例如: `2014/300 13:47:11.954796`

`-T fields|pdml|ps|psml|text` 设定打印解析后的包数据时的输出格式。选项可以是下面之一：

`fields` 以字段的形式打印结果，关于具体 `fields` 的值是通过 `-e` 选项来指定的，以及 `-E` 选项指定打印的形态。  
例如: `-T fields -E separator=, -E quote=d -e http.host -e http.request.uri`  
指定了以逗号分隔，并用双引号包裹字段值，只打印 `http` 协议的 `host` 与 `request.uri` 的信息。

`pdml` Packet Details Markup Language 数据包详细信息标记语言，针对解析完的数据包详细信息的一种基于XML的格式。  
这些信息跟使用 `-v` 选项打印出来的包详细信息是一致的。

`ps` PostScript 人类可读的针对每个包一行总结信息，或者针对每个包的多行详细信息，依赖于 `-v` 选项是不是给出。

`psml` Packet Summary Markup Language 数据包汇总信息标记语言，针对解析完的数据包汇总信息的一种基于XML的格式。  
这些信息跟不带选项抓包，默认打印出来的，每行针对每个数据包汇总信息是一样的。

`text` 人类可读的每个数据包一行汇总信息的文本格式，或者针对每个数据包的多行详细信息，依赖于 `-v` 选项是不是给出。

`-u <seconds type>` 指定秒格式，可用选项是：

- `s` 代表秒
- `hms` 代表 小时, 分钟, 秒

`-v` 打印版本信息，并退出。

`-w <outfile>|-` 指将捕获到的“生包(raw packet)”保存到 `outfile` 文件，如果 `outfile` 等于 `-` 的话，会将结果导入到标准输出。  
注意，`-w` 指定保存“生包”不是文本文件。

`-W <file format option>` 如果文件格式支持的话，保存额外的信息到文件。例如：

- `-F pcapng -W n`  
将会保存对于捕获到的包数据中 主机名的解析记录。  
将来版本的 Wireshark 可能会自动改变捕获的包文件格式为 `pcapng` 格式。  
这个选项的值是一个字符串，可能包含以下字符：  
`n` 写入网络地址的解析信息 (pcapng only)

`-x` 使 `tshark` 在打印完包的汇总或者详细信息后，还打印出包里面数据的 hex 和 ASCII 信息。

`-X <eXtension options>` 指定传给 `tshark` 的 `model` 的选项。  
该选项的格式为: `extension_key:value`, `extension_key` 的值可以是：  
`lua_script:lua_script_filename` 告诉 `tshark` 去加载一个除了默认的 `lua` 脚本之外的 `lua` 脚本。



**lua\_script****num:argument** 以 **num** 为标示符, 告诉 **tshark** 传递给定的参数到对应的 **lua\_script**。

**num** 以数字为索引, 代表 **lua\_script** 命令的顺序。

例如: 如果只有一个脚本通过 **-X lua\_script:my.lua** 命令加载, 那么参数 **-X lua\_script1:foo** 将会传递字符串 **foo** 到脚本 **my.lua**。如果有两个脚本通过命令 **-X lua\_script:my.lua** 和 **-X lua\_script:other.lua** 被加载, 那么 **-X lua\_script2:bar** 将会把字符串 **bar** 传递给 **other.lua**。

**read\_format:file\_format** 告诉 **tshark** 使用给定的文件格式读入之前捕获到的包数据文件 (文件通过 **-r** 选项给定)。

如果没有提供文件格式, 或者提供了一个不可用的文件格式, 将会导致产生一个可用格式的文件。

**-y <capture link type>** 设定捕获数据时使用的数据链路类型。候选值可以通过运行 **-L** 选项的结果来获取。

这个选项可以使用多次, 如果出现在第一个 **-i** 选项之前, 代表设定默认的捕获链路类型。否则设定的为特定网口的链路类型。

**-Y <display filter>** 设定 “读取过滤器表达式”, 类似于 **-R** 选项, 但是用在 **single-pass analysis**。如果使用 **two-pass analysis** (see **-2**) 只有匹配 **-R** 指定的 “过滤器表达式(如果有得话)” 的包, 才会再次被本过滤器检查。

该选项在解析捕获到的包或者将捕获到得包写入到文件之前应用 “读取过滤器”。匹配这个过滤器的包将会被打印或者写入到文件。被匹配包依赖的包 (e.g., **fragments**) 也会写入到文件, 但是不会被打印出来。既不匹配又不依赖的包将会被丢弃。

**-z <statistics>** 使 **tshark** 在捕获完成后, 或者读取捕获文件文件完成后, 展示各种统计信息。

可以配合 **-q** 选项一起使用, 这样就只查看统计信息, 而不受那些包信息的影响。

但这有一个例外的情况, 就是 **-z proto** 是不能和 **-q** 或者 **-v** 选项一起使用的, 具体看下面的介绍。

注意: 因为讲解这个选项时, 我使用的是 **tshark 1.0.15** 版本做的截图, 此版本还没有 **-y** 选项, 所以下面的命令把 **-y** 选项全部写做了 **-R** 选项。

**-z help** 展示所有 **-z** 选项可用的候选值列表。

因为 **-z help** 下有很多选项, 不同版本的 **tshark** 的选项也不相同, 这里只针对一些常见的有代表的选项做一下解释, 其他没有涉及到得选项请查看对应 **man** 帮助。

**-z conv,type[,filter]** 展示一个表格, 关于捕获到的会话 (**conersations**) 信息。 **type** 标记指定我们要进行统计的会话终端(endpoint)类型。当前支持的类型有:

```
"eth"      Ethernet addresses
"fc"       Fibre Channel addresses
"fddi"     FDDI addresses
"ip"       IPv4 addresses
"ipv6"     IPv6 addresses
"ipx"      IPX addresses
"tcp"      TCP/IP socket pairs  Both IPv4 and IPv6 are supported
"tr"       Token Ring addresses
"udp"      UDP/IP socket pairs  Both IPv4 and IPv6 are supported
```

如果过滤器 **filter** 标记也同时给出的话, 只有匹配这个过滤器的包会被计算和统计。

例子:

```
#tshark -c 500 -ni eth0 -f "port 80" -qz conv,eth
```

统计在eth0网口上捕获500个包, 不解析地址和端口, 并且只捕获 80 端口包的信息

```
Capturing on eth0
500 packets captured
=====
Ethernet Conversations
Filter:<No Filter>

| Frames | Bytes | | Frames | Bytes | | Total | | |
|-----|-----| |-----|-----| |-----|
| a4:ba:db:36:89:e6 | <-> | a4:ba:db:36:8e:be | 93 | 15600 | 81 | 40108 | 174 | 55708 |
| 00:1b:21:c9:37:78 | <-> | a4:ba:db:36:89:e6 | 59 | 5121 | 72 | 87892 | 131 | 93013 |
| a4:ba:db:36:89:e6 | <-> | a4:ba:db:36:a1:d8 | 27 | 12021 | 28 | 3117 | 55 | 15138 |
| a4:ba:db:36:89:e6 | <-> | a4:ba:db:36:97:c3 | 15 | 3932 | 17 | 1844 | 32 | 5776 |
| a4:ba:db:36:89:e6 | <-> | a4:ba:db:36:a5:10 | 14 | 2502 | 18 | 3564 | 32 | 6066 |
| 84:2b:2b:5e:56:f9 | <-> | a4:ba:db:36:89:e6 | 16 | 1928 | 14 | 7150 | 30 | 9078 |
| 00:1b:21:c9:36:68 | <-> | a4:ba:db:36:89:e6 | 13 | 1805 | 11 | 3366 | 24 | 5171 |
| a4:ba:db:36:89:e6 | <-> | a4:ba:db:36:98:e5 | 10 | 1046 | 12 | 2419 | 22 | 3465 |
=====
```

```
#tshark -c 100 -ni eth0 -f "port 80" -qz conv,ip,"ip.src==192.168.0.141"
```

统计在eth0网口上捕获100个包, 不解析地址和端口, 并且只捕获80端口而且源地址为192.168.0.141的包信息

```
Capturing on eth0
100 packets captured
=====
IPv4 Conversations
Filter:ip.src==192.168.0.141

| Frames | Bytes | | Frames | Bytes | | Total | | |
|-----|-----| |-----|-----| |-----|
| 192.168.0.141 | <-> | 192.168.0.123 | 0 | 0 | 11 | 726 | 11 | 726 |
| 192.168.0.141 | <-> | 192.168.0.22 | 0 | 0 | 7 | 736 | 7 | 736 |
| 192.168.0.141 | <-> | 192.168.0.23 | 0 | 0 | 6 | 1335 | 6 | 1335 |
=====
```

**-z http,stat**, 计算 HTTP 的统计分布信息。展示的值是 HTTP 的回应状态码, 和 HTTP 的请求方法。

例如:

```
tshark -c 1000 -ni eth0 -f "port 80" -qz http,stat,
```

```
Capturing on eth0
1000 packets captured

=====
HTTP Statistics
* HTTP Status Codes in reply packets
  HTTP 301 Moved Permanently
  HTTP 302 Moved Temporarily
  HTTP 304 Not Modified
  HTTP 200 OK
* List of HTTP Request methods
  GET 85
  POST 2
=====
```

-z **http,tree** 计算 HTTP 的统计分布信息。展示的值是 HTTP 的请求方法, 和 HTTP 的回应状态码。

例如:

```
tshark -c 1000 -ni eth0 -f "port 80" -qz http,tree
```

```
Capturing on eth0
1000 packets captured

=====
HTTP/Packet Counter      value      rate      percent
-----
Total HTTP Packets      180      2.019907
HTTP Request Packets    73      0.819185      40.56%
  GET                    73      0.819185      100.00%
HTTP Response Packets   46      0.516199      25.56%
  ??? : broken          0      0.000000      0.00%
  1xx : Informational    0      0.000000      0.00%
  2xx : Success          43      0.482533      93.48%
    200 OK               43      0.482533      100.00%
  3xx : Redirection      2      0.022443      4.35%
    301 Moved Permanently 2      0.022443      100.00%
  4xx : Client Error     1      0.011222      2.17%
    404 Not Found        1      0.011222      100.00%
  5xx : Server Error     0      0.000000      0.00%
  Other HTTP Packets     61      0.684524      33.89%
=====
```

-z **http\_req,tree** 统计HTTP到服务器的请求信息。显示的信息是请求的域名和URI路径。

例如:

```
tshark -c 100 -ni eth0 -f "port 80" -qz http_req,tree
```

```
Capturing on eth0
100 packets captured

=====
HTTP/Requests      value      rate      percent
-----
HTTP Requests by HTTP Host
  [redacted].cn      2      0.213812      20.00%
    /jiedaoxinxi/124663791.html 2      0.213812      100.00%
  m.[redacted].cn    2      0.213812      20.00%
    /63/xincheng/[redacted] 2      0.213812      100.00%
  weihai [redacted] 2      0.213812      20.00%
    /63/[redacted] 2      0.213812      100.00%
  m.[redacted].cn    2      0.213812      20.00%
    /c[redacted].php?catid=451&geoid=10917&pagename=m_item 2      0.213812      100.00%
  qingdao.[redacted].cn 2      0.213812      20.00%
    /jiedaoxinxi/96632343.html 2      0.213812      100.00%
=====
```

-z **http\_srv,tree** 跟上面类似, 但是统计的是回应信息。

例如:

```
tshark -c 100 -ni eth0 -f "port 80" -qz http_srv,tree
```

```
Capturing on eth0
100 packets captured

=====
HTTP/Load Distribution
=====
value      rate      percent
-----
HTTP Requests by Server      2      0.044412
HTTP Requests by Server Address      2      0.044412      100.00%
192.168.0.102      1      0.022206      50.00%
y. . . . .cn      1      0.022206      100.00%
192.168.0.24      1      0.022206      50.00%
y. . . . .cn      1      0.022206      100.00%
HTTP Requests by HTTP Host      2      0.044412      100.00%
yi. . . . .cn      2      0.044412      100.00%
192.168.0.102      1      0.022206      50.00%
192.168.0.24      1      0.022206      50.00%
HTTP Responses by Server Address      2      0.044412
192.168.0.23      1      0.022206      50.00%
OK      1      0.022206      100.00%
192.168.0.25      1      0.022206      50.00%
OK      1      0.022206      100.00%
=====
```

**-z io,phs[,filter]** 一个基于协议层次结构中数据包的个数和比特字节的统计表。如果**filter**给出的话，就只计算通过这个过滤器的包。

例如：

```
tshark -c 100 -ni eth0 -f "port 80" -qz io,phs
Capturing on eth0
100 packets captured

=====
Protocol Hierarchy Statistics
Filter: frame
=====
frame      frames:100 bytes:52143
eth      frames:100 bytes:52143
ip      frames:100 bytes:52143
tcp      frames:100 bytes:52143
http      frames:17 bytes:39540
png      frames:1 bytes:2974
unresembled      frames:1 bytes:2974
data-text-lines      frames:1 bytes:333
tcp.segments      frames:2 bytes:153
http      frames:2 bytes:153
png      frames:1 bytes:90
data-text-lines      frames:1 bytes:63
=====
```

**-z proto,colinfo,filter,field** 最后我们来说说，**-z** 选项里面一个“特立独行”的标记，因为它确切来说，不是关于统计信息，而是来在**tshark** 默认输出信息的样式后面添加指定的**field**信息，也正因为如此，它不能和 **-q** 与 **-v** 选项一起使用，不然就看不到了嘛。

举例来说，首先看看 **tshark** 的关于 **http** 协议中 **get** 请求的默认输出信息的样式：

```
# tshark -c 100 -ni eth0 -f "port 80" -R http.request
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
0.000006 192.168.0.102 -> 192.168.0.102 HTTP GET /63/nankaizhoubian/zhuzhaichuzu/3/ HTTP/1.1
0.000173 192.168.0.141 -> 192.168.0.123 HTTP GET /63/nankaizhoubian/zhuzhaichuzu/3/ HTTP/1.1
0.012993 180 -> 192.168.0.102 HTTP GET /ui/templates/html/js/featured_ads.js?t=4&g=6927&c=246
0.019884 180 -> 192.168.0.102 HTTP GET /ui/templates/html/js/featured_ads.js?t=3&g=6927&c=246
0.020798 192.168.0.141 -> 192.168.0.22 HTTP GET /ui/templates/html/js/featured_ads.js?t=3&g=6927&c=246
0.020876 192.168.0.141 -> 192.168.0.20 HTTP GET /ui/templates/html/js/featured_ads.js?t=4&g=6927&c=246
0.021219 192.168.0.141 -> 192.168.0.102 HTTP GET /ui/templates/html/js/featured_ads.js?t=3&g=194&c=253&g
0.021428 192.168.0.141 -> 192.168.0.24 HTTP GET /ui/templates/html/js/featured_ads.js?t=3&g=194&c=253&g
0.022020 192.168.0.141 -> 192.168.0.26 HTTP POST /ui/templates/html/js/featured_ads.js?cmd=insert&geo=416&cat=738 HTTP
9 packets captured
```

这个是 **tshark** 对数据包解析后的默认的输出样式。

如果这个输出样式让你感觉不足的话，例如，你想要看到数据包里**http**协议的**User-Agent**的信息，你有两个选择：

第一个是像上面介绍的那样，使用 **-T -e -B** 选项这套组合拳来修改输出样式，定制任意你想得到的样式，例如：

```
tshark -c 100 -ni eth0 -f "port 80" -R http.request -T fields -E quote=d -e http.request.method -e
http.request.uri -e http.user_agent
```

得到下面的格式：



```

Capturing on eth0
"GET" "/jiedaoxinxi/67654782.html" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/"
"GET" "/jiedaoxinxi/67654782.html" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/"
"GET" "/73/huangshan/xiaoxingquan/jiwawagou/" "Sogou web spider/4.0(+http://www.sogou.com/docs/help/we
"GET" "/73/huangshan/xiaoxingquan/jiwawagou/" "Sogou web spider/4.0(+http://www.sogou.com/docs/help/we
"GET" "/lianxi/46541963.html" "Mozilla/5.0 (Linux;u;Android 2.3.7;zh-cn;) AppleWebKit/533.1 (KHTML,lik
5 packets captured

```

第二个就是使用 **-z proto,colinfo,filter,field** 来实现。

例如：

```

tshark -c 100 -ni eth0 -f "port 80" -R http.request -z
proto,colinfo,http.user_agent,http.user_agent

```

```

Capturing on eth0
0.000013 -> 192.168.0.102 HTTP GET /...?pagename=m_ge
user_agent == "Mozilla/5.0 (Linux; U; Android 4.0.2; en-us; Galaxy Nexus Build/ICL53F) AppleWebKit/534.3
sion/4.0 Mobile Safari/534.30"
0.000064 192.168.0.141 -> 192.168.0.24 HTTP GET /g...?pagename=m_geo_
er_agent == "Mozilla/5.0 (Linux; U; Android 4.0.2; en-us; Galaxy Nexus Build/ICL53F) AppleWebKit/534.30
on/4.0 Mobile Safari/534.30"
2 packets captured

```

现在满足你得需求了。

另外， 此选项可以使用多次，添加多个字段到你汇总信息中。例如：

```

tshark -c 100 -ni eth0 -f "port 80" -R http.request -z "proto,colinfo,http.host,http.host" -z
"proto,colinfo,http.request,http.request.method"

```

**--capture-comment <comment>** 添加捕获备注到捕获的文件里。

这个选项值能再被保存的文件格式是 **pcapng** 时才可用。

给个保存的文件，只能添加一个备注。

例子：

```

tshark -ni en0 -f "tcp" -Y "http" 捕获本机 en0 网卡上的 tcp 连接数据，并且不DNS/Port翻译 (-n)，但是只展
示 http 协议的内容。

```