10.2. Start Wireshark from the command line

You can start Wireshark from the command line, but it can also be started from most Window managers as well. In this section we will look at starting it from the command line.

Wireshark supports a large number of command line parameters. To see what they are, simply enter the command *wireshark -h* and the help information shown in Example 10.1, "Help information available from Wireshark" (or something similar) should be printed.

Example 10.1. Help information available from Wireshark

```
Wireshark 1.12.1 (Git Rev Unknown from unknown)
Interactively dump and analyze network traffic.
See https://www.wireshark.org for more information.

Copyright 1998-2014 Gerald Combs <gerald@wireshark.org> and contributors.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Usage: wireshark [options] ... [ <infile> ]

Capture interface:
  -i <interface>           name or idx of interface (def: first non-loopback)
  -f <capture filter>      packet filter in libpcap filter syntax
  -s <snaplen>             packet snapshot length (def: 65535)
  -p                       don't capture in promiscuous mode
  -k                       start capturing immediately (def: do nothing)
  -S                       update packet display when new packets are captured
  -l                       turn on automatic scrolling while -S is in use
  -I                       capture in monitor mode, if available
  -B <buffer size>         size of kernel buffer (def: 2MB)
  -y <link type>           link layer type (def: first appropriate)
  -D                       print list of interfaces and exit
  -L                       print list of link-layer types of iface and exit

Capture stop conditions:
  -c <packet count>        stop after n packets (def: infinite)
  -a <autostop cond.> ...  duration:NUM - stop after NUM seconds
                           filesize:NUM - stop this file after NUM KB
                              files:NUM - stop after NUM files
Capture output:
  -b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                           filesize:NUM - switch to next file after NUM KB
                              files:NUM - ringbuffer: replace after NUM files
Input file:
  -r <infile>              set the filename to read from (no pipes or stdin!)

Processing:
  -R <read filter>         packet filter in Wireshark display filter syntax
  -n                       disable all name resolutions (def: all enabled)
  -N <name resolve flags>  enable specific name resolution(s): "mntC"

User interface:
  -C <config profile>      start with specified configuration profile
```

```
  -Y <display filter>      start with the given display filter
  -g <packet number>       go to specified packet number after "-r"
  -J <jump filter>         jump to the first packet matching the (display)
                           filter
  -j                       search backwards for a matching packet after "-J"
  -m <font>                set the font name used for most text
  -t a|ad|d|dd|e|r|u|ud    output format of time stamps (def: r: rel. to first)
  -u s|hms                 output format of seconds (def: s: seconds)
  -X <key>:<value>         eXtension options, see man page for details
  -z <statistics>          show various statistics, see man page for details

Output:
  -w <outfile|->           set the output filename (or '-' for stdout)

Miscellaneous:
  -h                       display this help and exit
  -v                       display version info and exit
  -P <key>:<path>          persconf:path - personal configuration files
                           persdata:path - personal data files
  -o <name>:<value> ...    override preference or recent setting
  -K <keytab>              keytab file to use for kerberos decryption
  --display=DISPLAY        X display to use
```

We will examine each of the command line options in turn.

The first thing to notice is that issuing the command `wireshark` by itself will bring up Wireshark. However, you can include as many of the command line parameters as you like. Their meanings are as follows (in alphabetical order):

-a <capture autostop condition>

Specify a criterion that specifies when Wireshark is to stop writing to a capture file. The criterion is of the form test:value, where test is one of:

duration:value

Stop writing to a capture file after value of seconds have elapsed.

filesize:value

Stop writing to a capture file after it reaches a size of value kilobytes (where a kilobyte is 1000 bytes, not 1024 bytes). If this option is used together with the -b option, Wireshark will stop writing to the current capture file and switch to the next one if filesize is reached.

files:value

Stop writing to capture files after value number of files were written.

-b <capture ring buffer option>

If a maximum capture file size was specified, this option causes Wireshark to run in “ring buffer” mode, with the specified number of files. In “ring buffer” mode, Wireshark will write to several capture files. Their name is based on the number of the file and on the creation date and time.

When the first capture file fills up Wireshark will switch to writing to the next file, and so on. With the <command>files</command> option it’s also possible to form a “ring buffer.” This will fill up new files until the number of files specified, at which point the data in the first file will be discarded so a new file can be written.

If the optional <command>duration</command> is specified, Wireshark will also switch to the next file when the specified number of seconds has elapsed even if the current file is not completely fills up.

duration</command>:value
Switch to the next file after value seconds have elapsed, even if the current file is not completely filled up.

filesize</command>:value
Switch to the next file after it reaches a size of value kilobytes (where a kilobyte is 1000 bytes, not 1024 bytes).

files</command>:value
Begin again with the first file after value number of files were written (form a ring buffer).

-B <capture buffer size>
Set capture buffer size (in MB, default is 1MB). This is used by the capture driver to buffer packet data until that data can be written to disk. If you encounter packet drops while capturing, try to increase this size. Not supported on some platforms.

-c <capture packet count>
This option specifies the maximum number of packets to capture when capturing live data. It would be used in conjunction with the `-k` option.

-D
Print a list of the interfaces on which Wireshark can capture, then exit. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the `-i` flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking `ifconfig -a`). The number can be especially useful on Windows, where the interface name is a GUID.

Note that "can capture" means that Wireshark was able to open that device to do a live capture. If, on your system, a program doing a network capture must be run from an account with special privileges (for example, as root), then, if Wireshark is run with the `-D` flag and is not run from such an account, it will not list any interfaces.

-f <capture filter>
This option sets the initial capture filter expression to be used when capturing packets.

-g <packet number>
After reading in a capture file using the -r flag, go to the given packet number.

-h
The `-h` option requests Wireshark to print its version and usage instructions (as shown above) and exit.

-i <capture interface>
Set the name of the network interface or pipe to use for live packet capture.

Network interface names should match one of the names listed in `wireshark -D` (described above). A number, as reported by `wireshark -D`, can also be used. If you're using UNIX, `netstat -i` or `ifconfig -a` might also work to list interface names, although not all versions of UNIX support the `-a` flag to `ifconfig`.

If no interface is specified, Wireshark searches the list of interfaces, choosing the first non-loopback interface if there are any non-loopback interfaces, and choosing the first loopback

interface if there are no non-loopback interfaces; if there are no interfaces, Wireshark reports an error and doesn't start the capture.

Pipe names should be either the name of a FIFO (named pipe) or "-" to read data from the standard input. Data read from pipes must be in standard libpcap format.

-J <jump filter>
After reading in a capture file using the `-r` flag, jump to the first packet which matches the filter expression. The filter expression is in display filter format. If an exact match cannot be found the first packet afterwards is selected.

-I
Capture wireless packets in monitor mode if available.

-j
Use this option after the `-J` option to search backwards for a first packet to go to.

-k
The `-k` option specifies that Wireshark should start capturing packets immediately. This option requires the use of the `-i` parameter to specify the interface that packet capture will occur from.

-K <keytab file>
Use the specified file for Kerberos decryption.

-l
This option turns on automatic scrolling if the packet list pane is being updated automatically as packets arrive during a capture (as specified by the `-S` flag).

-L
List the data link types supported by the interface and exit.

-m <font>
This option sets the name of the font used for most text displayed by Wireshark.

-n
Disable network object name resolution (such as hostname, TCP and UDP port names).

-N <name resolving flags>
Turns on name resolving for particular types of addresses and port numbers. The argument is a string that may contain the letters `m` to enable MAC address resolution, `n` to enable network address resolution, and `t` to enable transport-layer port number resolution. This overrides `-n` if both `-N` and `-n` are present. The letter C enables concurrent (asynchronous) DNS lookups.

-o <preference or recent settings>
Sets a preference or recent value, overriding the default value and any value read from a preference or recent file. The argument to the flag is a string of the form *prefname:value*, where *prefname* is the name of the preference (which is the same name that would appear in the `preferences` or `recent` file), and *value* is the value to which it should be set. Multiple instances of `-o <preference settings> ` can be given on a single command line.

An example of setting a single preference would be:

```
wireshark -o mgcp.display_dissect_tree:TRUE
```

An example of setting multiple preferences would be:

```
wireshark -o mgcp.display_dissect_tree:TRUE -o mgcp.udp.callagent_port:2627
```

You can get a list of all available preference strings from the preferences file. See Appendix B, *Files and Folders* for details.

User access tables can be overridden using "uat," followed by the UAT file name and a valid record for the file:

```
wireshark -o "uat:user_dlts:\"User 0 (DLT=147)\",\"http\",\"0\",\"\",\"0\",\""
```

The example above would dissect packets with a libpcap data link type 147 as HTTP, just as if you had configured it in the DLT_USER protocol preferences.

-p

Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason. Hence, `-p` cannot be used to ensure that the only traffic that is captured is traffic sent to or from the machine on which Wireshark is running, broadcast traffic, and multicast traffic to addresses received by that machine.

-P <path setting>

Special path settings usually detected automatically. This is used for special cases, e.g. starting Wireshark from a known location on an USB stick.

The criterion is of the form key:path, where key is one of:

persconf:path

Path of personal configuration files, like the preferences files.

persdata:path

Path of personal data files, it's the folder initially opened. After the initialization, the recent file will keep the folder last used.

-Q

This option forces Wireshark to exit when capturing is complete. It can be used with the `-c` option. It must be used in conjunction with the `-i` and `-w` options.

-r <infile>

This option provides the name of a capture file for Wireshark to read and display. This capture file can be in one of the formats Wireshark understands.

-R <read (display) filter>

This option specifies a display filter to be applied when reading packets from a capture file. The syntax of this filter is that of the display filters discussed in Section 6.3, "Filtering packets while viewing". Packets not matching the filter are discarded.

-s <capture snapshot length>

This option specifies the snapshot length to use when capturing packets. Wireshark will only capture *snaplen* bytes of data for each packet.

-S

This option specifies that Wireshark will display packets as it captures them. This is done by capturing in one process and displaying them in a separate process. This is the same as "Update list of packets in real time" in the "Capture Options" dialog box.

-t <time stamp format>

This option sets the format of packet timestamps that are displayed in the packet list window. The format can be one of:

r

Relative, which specifies timestamps are displayed relative to the first packet captured.

a

Absolute, which specifies that actual times be displayed for all packets.

ad

Absolute with date, which specifies that actual dates and times be displayed for all

packets.

d
Delta, which specifies that timestamps are relative to the previous packet.

e
Epoch, which specifies that timestamps are seconds since epoch (Jan 1, 1970 00:00:00)

-u <s | hms>
Show timesamps as seconds (*s*, the default) or hours, minutes, and seconts (*hms*)

-v
The `-v` option requests Wireshark to print out its version information and exit.

-w <savefile>
This option sets the name of the file to be used to save captured packets.

-y <capture link type>
If a capture is started from the command line with -k, set the data link type to use while capturing packets. The values reported by -L are the values that can be used.

-X <eXtension option>
Specify an option to be passed to a TShark module. The eXtension option is in the form extension_key:value, where extension_key can be:

lua_script:lua_script_filename
Tells Wireshark to load the given script in addition to the default Lua scripts.

lua_script[num]:argument
Tells Wireshark to pass the given argument to the lua script identified by *num*, which is the number indexed order of the *lua_script* command. For example, if only one script was loaded with `-X lua_script:my.lua`, then `-X lua_script1:foo` will pass the string *foo* to the *my.lua* script. If two scripts were loaded, such as `-X lua_script:my.lua` and `-X lua_script:other.lua` in that order, then a `-X lua_script2:bar` would pass the string *bar* to the second lua script, namely *other.lua*.

-z <statistics-string>
Get Wireshark to collect various types of statistics and display the result in a window that updates in semi-real time.