4

WORKING WITH CAPTURED PACKETS

Now that you've been introduced to Wireshark, you're ready to start capturing and analyzing packets. In this chapter, you'll learn how to work with capture files, packets, and time-display formats. We'll also cover more advanced options for capturing packets and dive into the world of filters.

Working with Capture Files

You'll find that a good portion of your packet analysis will happen after your capture. Usually, you'll perform several captures at various times, save them, and analyze them all at once. Therefore, Wireshark allows you to save your capture files to be analyzed later. You can also merge multiple capture files.

Saving and Exporting Capture Files

To save a packet capture, select **File ▸ Save As**. You should see the Save file as dialog, as shown in Figure 4-1. You'll be asked for a location to save your packet capture and for the file format you wish to use. If you don't specify a file format, Wireshark will use the default *.pcapng* file format.

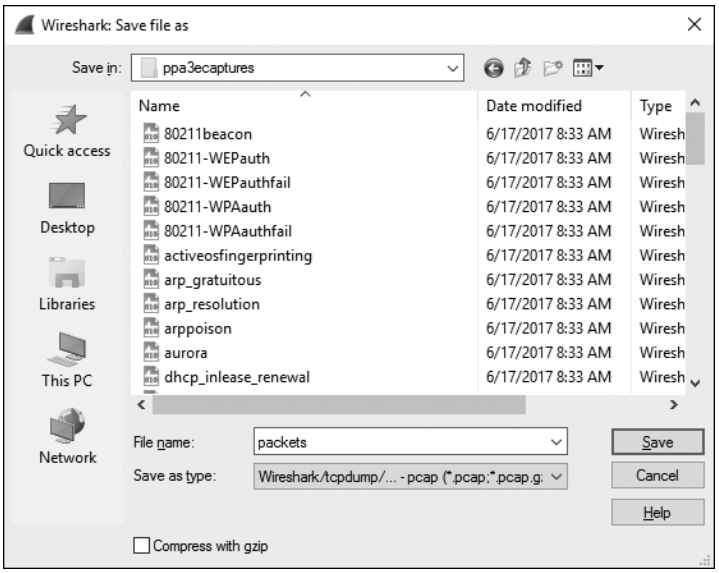


Figure 4-1: The Save file as dialog allows you to save your packet captures.

In many cases, you may only want to save a subset of the packets in your capture. To do so, select **File ▸ Export Specified Packets**. The dialog that appears is shown in Figure 4-2. This is a great way to thin bloated packet-capture files. You can choose to save only packets in a specific number range, marked packets, or packets visible as the result of a display filter (marked packets and filters are discussed later in this chapter).

You can export your Wireshark capture data into several formats for viewing in other media or for importing into other packet analysis tools. Formats include plaintext, PostScript, comma-separated values (CSV), and XML. To export your packet capture in one of these formats, choose **File ▸ Export Packet Dissections** and then select the format for the exported file. You'll see a Save As dialog containing options related to the format you've chosen.

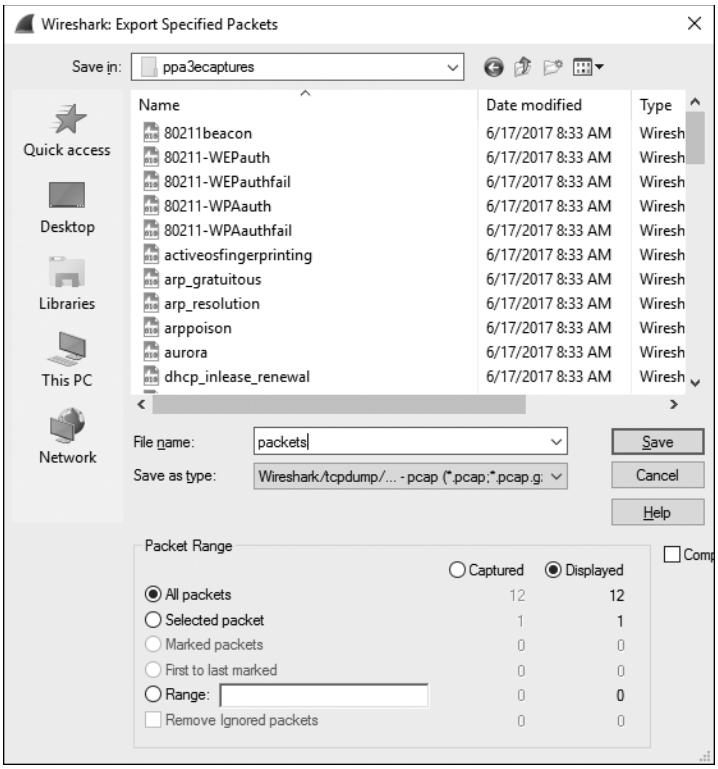


Figure 4-2: The Export Specified Packets dialog allows you to have more granular control over the packets you choose to save.

Merging Capture Files

Certain types of analysis require the ability to merge multiple capture files. This is a common practice when comparing two data streams or combining streams of the same traffic that were captured separately.

To merge capture files, open one of the files you want to merge and choose **File ▸ Merge** to bring up the Merge with capture file dialog, shown in Figure 4-3. Select the new file you wish to merge into the already open file and then select the method to use for merging the files. You can prepend the selected file to the currently open one, append it, or merge the files chronologically based on their timestamps.

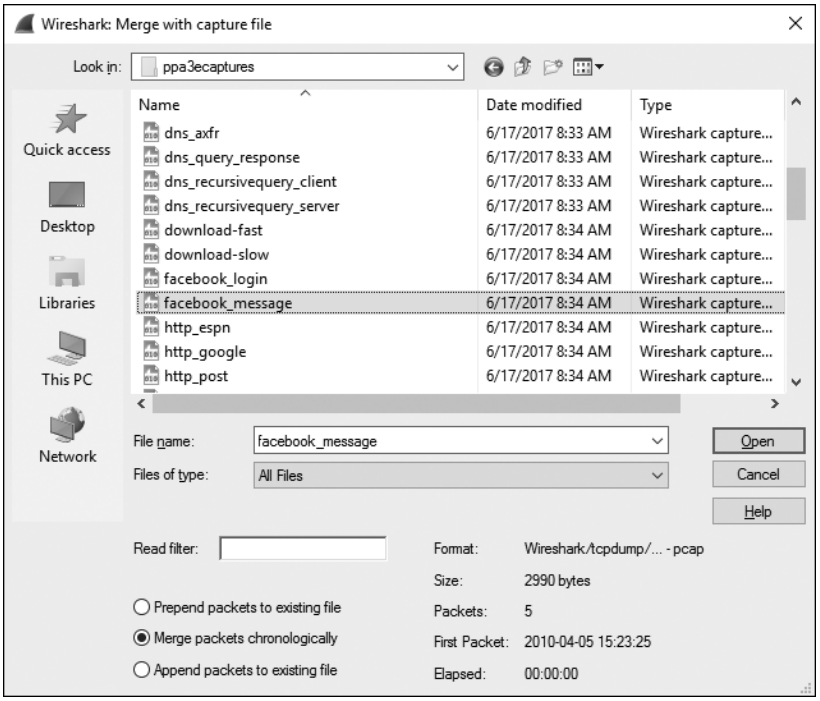


Figure 4-3: The Merge with capture file dialog allows you to merge two capture files.

Working with Packets

You will eventually encounter a situation involving a very large number of packets. As the number of packets grows into the thousands and even millions, you will need to navigate through packets more efficiently. For this purpose, Wireshark allows you to find and mark packets that match certain criteria. You can also print packets for easy reference.

Finding Packets

To find packets that match particular criteria, open the Find Packet bar, shown circled in Figure 4-4, by pressing CTRL-F. This bar should appear between the Filter bar and the Packet List pane.

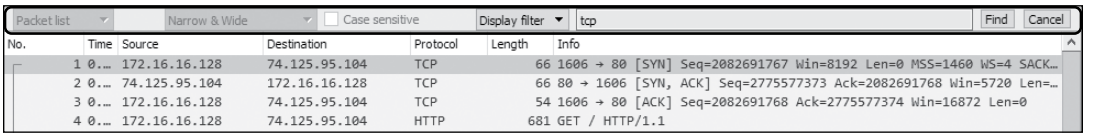


Figure 4-4: Finding packets in Wireshark based on specified criteria—in this case, packets matching the display filter expression `tcp`

This pane offers three options for finding packets:

- The Display filter option allows you to enter an expression-based filter that will find only those packets that satisfy that expression. This option is used in Figure 4-4.
- The Hex value option searches for packets with a hexadecimal value you specify.
- The String option searches for packets with a text string you specify. You can specify the pane the search is performed in or make the search string case sensitive.

Table 4-1 shows examples of these search types.

Table 4-1: Search Types for Finding Packets

Search type	Examples

Display filter	`not ip`
	`ip.addr==192.168.0.1`
	`arp`

Hex value	`00ff`
	`ffff`
	`00ABB1f0`

String	`Workstation1`
	`UserB`
	`domain`

Once you've decided which search type you will use, enter your search criteria in the text box and click **Find** to find the first packet that meets your criterion. To find the next matching packet, click **Find** again or press CTRL-N; find the previous matching packet by pressing CTRL-B.

Marking Packets

After you have found packets that match your criterion, you can mark those of particular interest. For example, marking packets will let you save only these packets. Also, you can find your marked packets quickly by their black background and white text, as shown in Figure 4-5.

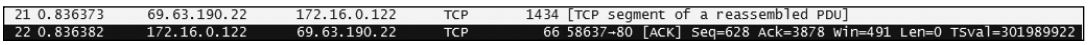


Figure 4-5: A marked packet is highlighted on your screen. In this example, the second packet is marked and appears darker.

To mark a packet, either right-click it in the Packet List pane and choose **Mark Packet** from the pop-up or click a packet in the Packet List pane and press CTRL-M. To unmark a packet, toggle this setting off by pressing CTRL-M again. You can mark as many packets as you wish in a capture. To jump forward and backward between marked packets, press SHIFT-CTRL-N and SHIFT-CTRL-B, respectively.

Printing Packets

Although most analysis will take place on the computer screen, you may need to print captured data. I occasionally print out packets and tape them to my desk so I can quickly reference their contents while doing other analysis. Being able to print packets to a PDF file is also very convenient, especially when preparing reports.

To print captured packets, open the Print dialog by choosing **File ▸ Print** from the main menu, as shown in Figure 4-6.

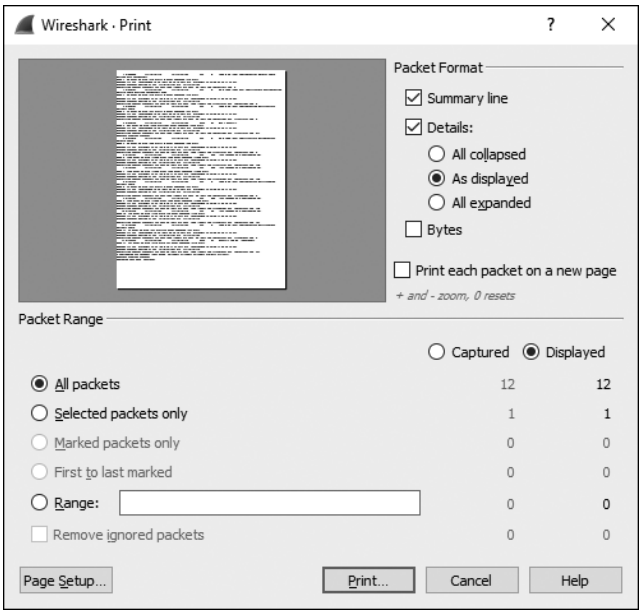


Figure 4-6: The Print dialog allows you to print the packets you specify.

As with the Export Specified Packets dialog, you can print a specific packet range, marked packets only, or packets displayed as the result of a filter. You can also select the level of detail you wish to print for each packet. Once you have selected the options, click **Print**.

Setting Time Display Formats and References

Time is of the essence—especially in packet analysis. Everything that happens on a network is time sensitive, and you will need to examine trends and network latency in capture files frequently. Wireshark supplies several configurable options related to time. In this section, we'll look at time display formats and references.

Time Display Formats

Each packet that Wireshark captures is given a timestamp, which is applied to the packet by the operating system. Wireshark can show the absolute timestamp, which indicates the exact moment when the packet was captured, as well as the time in relation to the last captured packet and the beginning and end of the capture.

Options related to time display are found under the View heading on the main menu. The Time Display Format section, shown in Figure 4-7, lets you configure the presentation format as well as the precision of the time display.

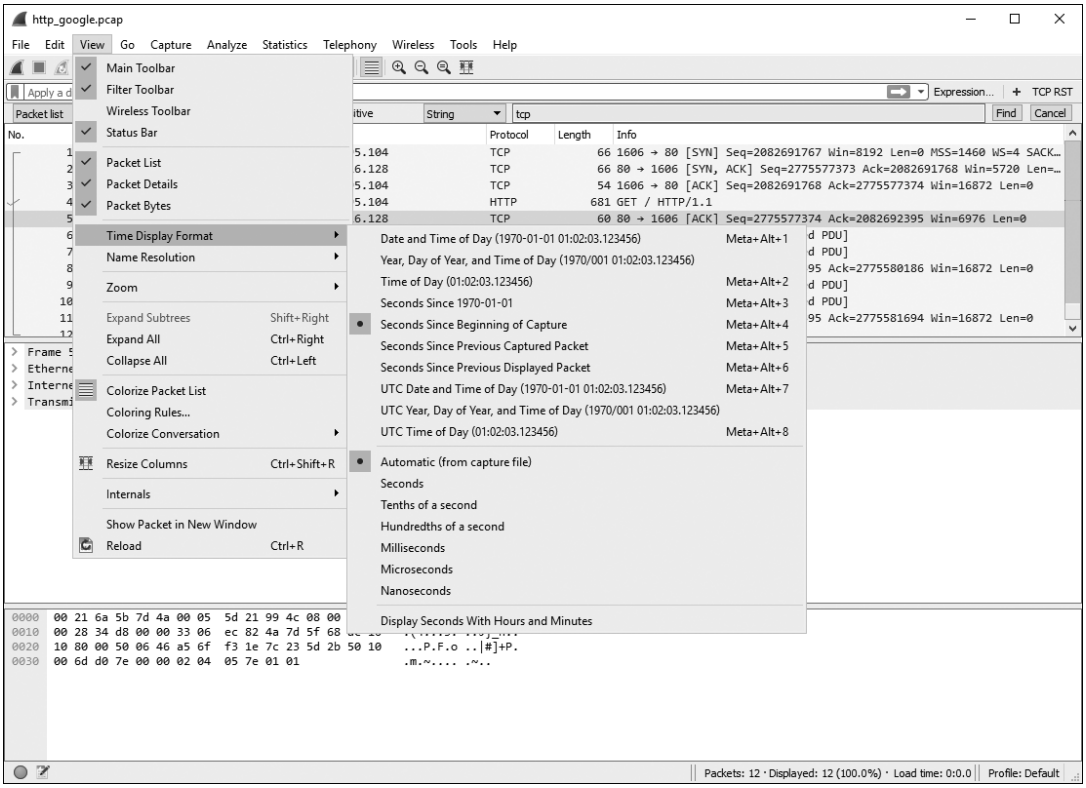


Figure 4-7: Several time display formats are available.

The presentation format options let you choose various settings for time display. These include date and time of day, UTC date and time of day, seconds since epoch, seconds since beginning of capture (the default setting), seconds since previous captured packet, and more.

The precision options allow you to set the time display precision to an automatic setting, which takes the format from the capture file, or to a manual setting, such as seconds, milliseconds, microseconds, and so on. We will be changing these options later in the book, so you should familiarize yourself with them now.

NOTE *When comparing packet data from multiple devices, be sure that the devices are synchronized with the same time source, especially if you are performing forensic analysis or troubleshooting. You can use the Network Time Protocol (NTP) to ensure network devices are synced. When examining packets from devices spanning more than one time zone, consider analyzing packets in UTC instead of local time to avoid confusion when reporting your findings.*

Packet Time Referencing

Packet time referencing allows you to configure a certain packet so that all subsequent time calculations are done in relation to that packet. This feature is particularly handy when you are examining a series of sequential events that are triggered at some point other than the start of the capture file.

To set a time reference to a packet, right-click the reference packet in the Packet List pane and choose **Set/Unset Time Reference**. To toggle this reference off, repeat the same action. You can also toggle a packet as a time reference on and off by selecting the packet you wish to reference in the Packet List pane and pressing CTRL-T.

When you enable a time reference on a packet, the Time column in the Packet List pane will display `*REF*`, as shown in Figure 4-8.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.030107	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406…
3	0.030182	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	*REF*	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.048778	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.070954	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
7	0.071217	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
8	0.071247	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775580186 Win=16872 Len=0

Figure 4-8: Packet 4 with the packet time reference toggle enabled

Setting a packet time reference is useful only when the time display format of a capture is set to display the time in relation to the beginning of the capture. Any other setting will produce no usable results and indeed will generate a set of times that can be very confusing.

Time Shifting

In some cases, you might encounter packets from multiple sources that are not synchronized to the same time source. This is especially common when examining capture files taken from two locations that contain the same stream of data. While most administrators desire a state in which every device on their network is synced, it's not uncommon for there to be a few seconds of time skew between certain types of devices. Wireshark provides the ability to shift the timestamp on packets to alleviate this problem during your analysis.

To shift the timestamp on one or more packets, select **Edit ▸ Time Shift** or press CTRL-SHIFT-T. On the Time Shift screen that opens, you can specify a time range to shift the entire capture file by, or you can specify a time to set individual packets to. In the example shown in Figure 4-9, I've chosen to shift the timestamp of every packet in the capture by adding two minutes and five seconds to each packet.

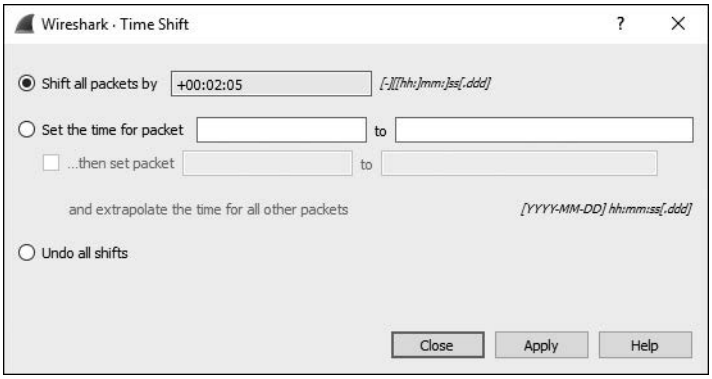


Figure 4-9: The Time Shift dialog

Setting Capture Options

We looked at the Capture Interfaces dialog while walking through a very basic packet capture in the last chapter. Wireshark offers quite a few additional capture options that we didn't address then. To access these options, choose **Capture ▸ Options**.

The Capture Interfaces dialog has a lot of bells and whistles, all designed to give you more flexibility while capturing packets. It's divided into three tabs: Input, Output, and Options. We'll examine each separately.

Input Tab

The main purpose of the Input tab (Figure 4-10) is to display all the interfaces available for capturing packets and some basic information for each interface. This includes the friendly name of the interface provided by the operating system, a traffic graph showing the throughput on the interface, and additional configuration options such as promiscuous mode status and buffer size. At the far right (not pictured), there is also a column for the applied capture filter, which we'll talk about in "Capture Filters" on page 65.

In this section, you can click most of these options and edit them inline. For example, if you want to disable promiscuous mode on an interface, you can click that field and change it from enabled to disabled via the provided drop-down menu.

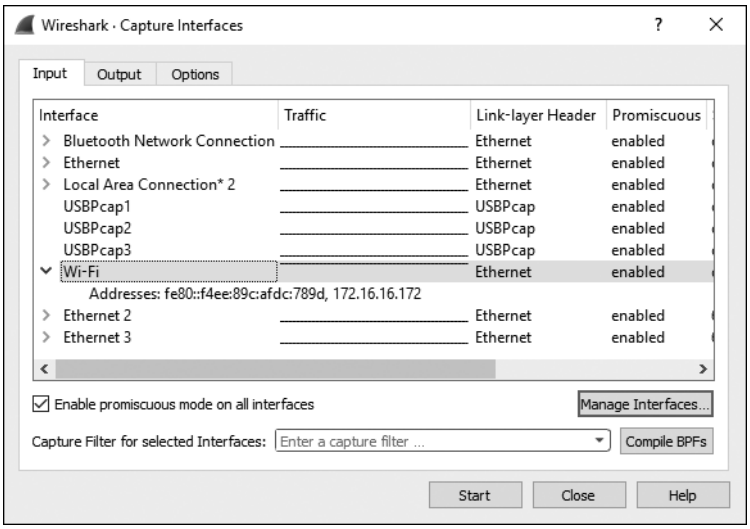


Figure 4-10: The Capture Interfaces Input options tab

Output Tab

The Output tab (Figure 4-11) allows you to automatically store captured packets in a file, rather than capturing them first and then saving the file. Doing so offers you more flexibility in managing how packets are saved. You can choose to save them as a single file or a file set or even use a ring buffer (which we'll cover in a moment) to manage the number of files created. To enable this option, enter a complete file path and name in the File text box. Alternatively, use the Browse... button to select a directory and provide a filename.

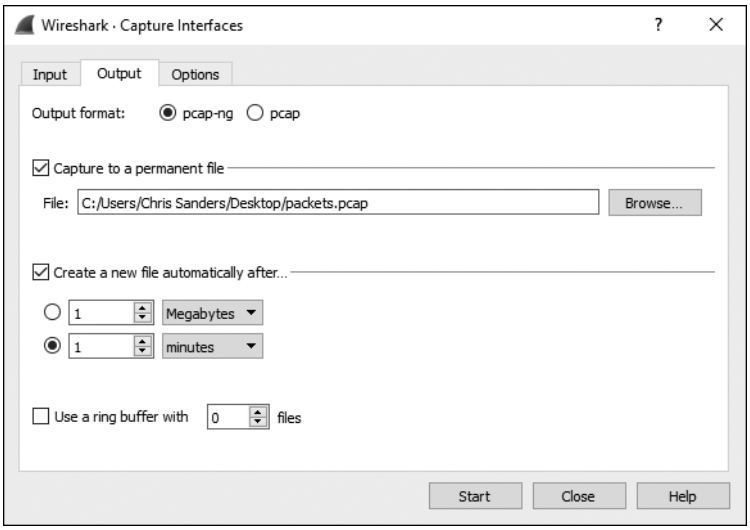
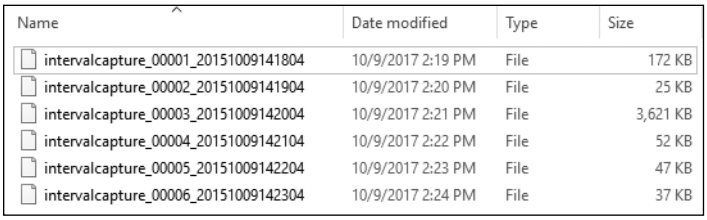


Figure 4-11: The Capture Interfaces Output options tab

When you are capturing a large amount of traffic or performing long-term captures, file sets can prove particularly useful. A *file set* is a grouping of multiple files separated by a particular condition. To save to a file set, check the **Create a new file automatically after...** option.

Wireshark uses various triggers to manage saving to file sets based upon a file size or time condition. To enable one of these triggers, select the radio button next to the size- or time-based option and then specify the value and unit on which to trigger. For instance, you can set a trigger that creates a new file after every 1MB of traffic captured or, as shown in Figure 4-12, after every minute of traffic captured.



Name	Date modified	Type	Size
intervalcapture_00001_20151009141804	10/9/2017 2:19 PM	File	172 KB
intervalcapture_00002_20151009141904	10/9/2017 2:20 PM	File	25 KB
intervalcapture_00003_20151009142004	10/9/2017 2:21 PM	File	3,621 KB
intervalcapture_00004_20151009142104	10/9/2017 2:22 PM	File	52 KB
intervalcapture_00005_20151009142204	10/9/2017 2:23 PM	File	47 KB
intervalcapture_00006_20151009142304	10/9/2017 2:24 PM	File	37 KB

Figure 4-12: A file set created by Wireshark at one-minute intervals

The Use a ring buffer option lets you specify a certain number of files your file set will hold before Wireshark begins to overwrite files. Although the term *ring buffer* has multiple meanings, for our purposes, it is essentially a file set that specifies that once the last file it can hold has been written, when more data must be saved, the first file is overwritten. In other words, it establishes a first in, first out (FIFO) method of writing files. You can check this option and specify the maximum number of files you wish to cycle through. For example, say you choose to use multiple files for your capture with a new file created every hour, and you set your ring buffer to 6. Once the sixth file has been created, the ring buffer will cycle back around and overwrite the first file rather than create a seventh file. This ensures that no more than six files (or in this case, hours) of data will remain on your hard drive, while still allowing new data to be written.

Lastly, the Output tab also lets you specify whether to use the *.pcapng* file format. If you plan to interact with your saved packets using a tool that isn't capable of parsing *.pcapng*, you can select the traditional *.pcap* format.

Options Tab

The Options tab contains a number of other packet-capturing choices, including display, name resolution, and capture termination options, shown in Figure 4-13.

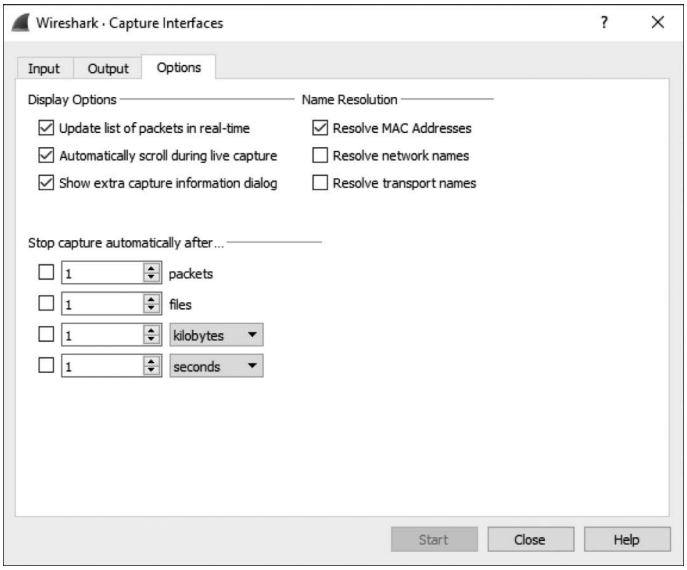


Figure 4-13: The Capture Interfaces Options tab

Display Options

The Display Options section controls how packets are shown as they are being captured. The Update list of packets in real-time option is self-explanatory and can be paired with the Automatically scroll during live capture option. When both of these options are enabled, all captured packets are displayed on the screen, with the most recently captured ones shown instantly.

WARNING *When paired, the Update list of packets in real-time and Automatically scroll during live capture options can be processor intensive, even when you are capturing a modest amount of data. Unless you have a specific need to see the packets in real time, it's best to deselect both options.*

The Show extra capture information dialog option lets you enable or suppress the display of a small window that shows the number and percentage of packets that have been captured, sorted by their protocol. I like to show the capture info dialog since I typically don't allow for the live scrolling of packets during capture.

Name Resolution Settings

The Name Resolution section options allow you to enable automatic MAC (layer 2), network (layer 3), and transport (layer 4) name resolution for your capture. We'll discuss name resolution as a general topic in more depth, including its drawbacks, in Chapter 5.

Stop Capture Settings

The Stop capture automatically after... section lets you stop the running capture when certain conditions are met. As with multiple file sets, you can trigger the capture to stop based on file size and time interval, but you can also trigger on number of packets. These options can be used with the multiple-file options on the Output tab.

Using Filters

Filters allow you to specify which packets you have available for analysis. Simply stated, a filter is an expression that defines criteria for the inclusion or exclusion of packets. If there are packets you don't want to see, you can write a filter that gets rid of them. If there are packets you want to see exclusively, you can write a filter that shows only those packets.

Wireshark offers two main types of filters:

- *Capture filters* are specified when packets are being captured and will capture only those packets that are specified for inclusion/exclusion in the given expression.
- *Display filters* are applied to an existing set of captured packets in order to hide unwanted packets or show desired packets based on the specified expression.

Let's look at capture filters first.

Capture Filters

Capture filters are applied during the packet-capturing process to limit the packets delivered to the analyst from the start. One primary reason for using a capture filter is performance. If you know that you do not need to analyze a particular form of traffic, you can simply filter it out with a capture filter and save the processing power that would typically be used in capturing those packets.

The ability to create custom capture filters comes in handy when you're dealing with large amounts of data. The analysis can be sped up by ensuring that you are looking at only the packet relevant to the issue at hand.

As an example, suppose you are troubleshooting an issue with a service running on port 262, but the server you are analyzing runs several different services on a variety of ports. Finding and analyzing only the traffic on one port would be quite a job in itself. To capture only the traffic on a specific port, you could use a capture filter. To do so, use the Capture Interfaces dialog as follows:

1. Choose the **Capture ▸ Options** button next to the interface on which you want to capture packets. This will open the Capture Interfaces dialog.
2. Find the interface you wish to use and scroll to the Capture Filter option in the far-right column.

3. You can apply the capture filter by clicking in this column to enter an expression. We want our filter to show only traffic inbound and outbound to port 262, so enter **port 262**, as shown in Figure 4-14. (We'll discuss expressions in more detail in the next section.) The color of the cell should turn green, indicating that you've entered a valid expression; it will turn red if the expression is invalid.

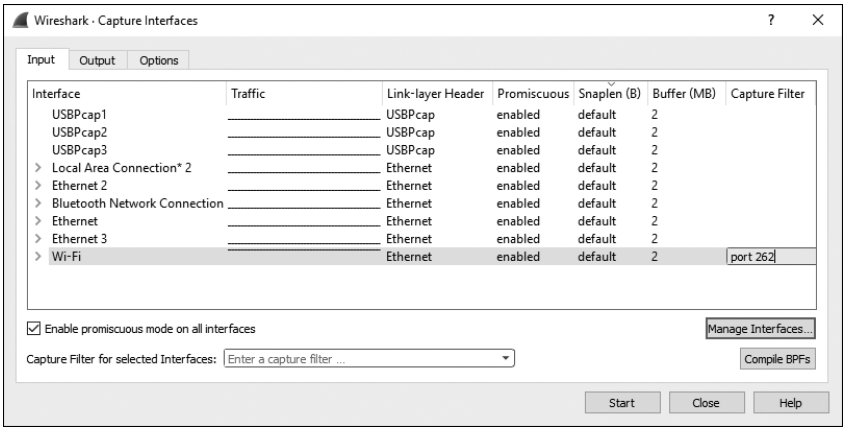


Figure 4-14: Creating a capture filter in the Capture Interfaces dialog

4. Once you have set your filter, click **Start** to begin the capture.

You should now see only port 262 traffic and be able to more efficiently analyze this particular data.

Capture/BPF Syntax

Capture filters are applied by libpcap/WinPcap and use the Berkeley Packet Filter (BPF) syntax. This syntax is common in several packet-sniffing applications, mostly because packet-sniffing applications tend to rely on the libpcap/WinPcap libraries, which allow for the use of BPFs. A knowledge of BPF syntax will be crucial as you dig deeper into networks at the packet level.

A filter created using the BPF syntax is called an *expression*, and each expression consists of one or more *primitives*. Primitives consist of one or more *qualifiers* (as listed in Table 4-2), followed by an ID name or number, as shown in Figure 4-15.

Table 4-2: The BPF Qualifiers

Qualifier	Description	Examples
Type	Identifies what the ID name or number refers to	`host`, `net`, `port`
Dir	Specifies a transfer direction to or from the ID name or number	`src`, `dst`
Proto	Restricts the match to a particular protocol	`ether`, `ip`, `tcp`, `udp`, `http`, `ftp`

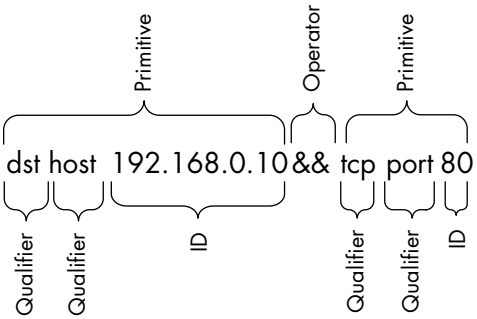


Figure 4-15: A sample capture filter

Given the components of an expression, a qualifier of `dst host` and an ID of 192.168.0.10 would combine to form a primitive. This primitive alone is an expression that would capture traffic only with a destination IP address of 192.168.0.10.

You can use logical operators to combine primitives to create more advanced expressions. Three logical operators are available:

- Concatenation operator AND (`&&`)
- Alternation operator OR (`||`)
- Negation operator NOT (`!`)

For example, the following expression will capture only traffic with a source IP address of 192.168.0.10 and a source or destination port of 80:

```
src host 192.168.0.10 && port 80
```

Hostname and Addressing Filters

Most filters you create will center on a particular network device or grouping of devices. Depending on the circumstances, filtering can be based on a device's MAC address, IPv4 address, IPv6 address, or DNS hostname.

For example, say you're curious about the traffic of a particular host that is interacting with a server on your network. From the server, you can create a filter using the `host` qualifier that captures all traffic associated with that host's IPv4 address:

```
host 172.16.16.149
```

If you are on an IPv6 network, you would filter based on an IPv6 address using the `host` qualifier, as shown here:

```
host 2001:db8:85a3::8a2e:370:7334
```

You can also filter based on a device's hostname with the `host` qualifier, like so:

```
host testserver2
```

Or, if you're concerned that the IP address for a host might change, you can filter based on its MAC address as well by adding the `ether` protocol qualifier:

```
ether host 00-1a-a0-52-e2-a0
```

The transfer direction qualifiers are often used in conjunction with filters, such as the ones in the previous examples, to capture traffic based on whether it's going to or coming from a host. For example, to capture only traffic coming from a particular host, add the `src` qualifier:

```
src host 172.16.16.149
```

To capture only data destined for 172.16.16.149, use the `dst` qualifier:

```
dst host 172.16.16.149
```

When you don't use a type qualifier (`host`, `net`, or `port`) with a primitive, the `host` qualifier is assumed. Therefore, this expression, which excludes that qualifier, is the equivalent of the preceding example:

```
dst 172.16.16.149
```

Port Filters

In addition to filtering on hosts, you can filter based on the ports used in each packet. Port filtering can be used to filter for services and applications that use known service ports. For example, here's a simple filter to capture traffic only to or from port 8080:

```
port 8080
```

To capture all traffic except that on port 8080, this would work:

```
!port 8080
```

The port filters can be combined with transfer direction qualifiers. For example, to capture only traffic going to the web server listening on the standard HTTP port 80, use the `dst` qualifier:

```
dst port 80
```

Protocol Filters

Protocol filters let you filter packets based on certain protocols. They are used to match non–application layer protocols that can't simply be defined by the use of a certain port. Thus, if you want to see only ICMP traffic, you could use this filter:

```
icmp
```

To see everything but IPv6 traffic, this will do the trick:

```
!ip6
```

Protocol Field Filters

One of the real strengths of the BPF syntax is the ability that it gives us to examine every byte of a protocol header in order to create very specific filters based on that data. The advanced filters that we'll discuss in this section will allow you to retrieve a specific number of bytes from a packet beginning at a particular location.

For example, suppose that we want to filter based on the type field of an ICMP header. The type field is located at the very beginning of a packet, which puts it at offset 0. To identify the location to examine within a packet, specify the byte offset in square brackets next to the protocol qualifier—`icmp[0]` in this example. This specification will return a 1-byte integer value that we can compare against. For instance, to get only ICMP packets that represent destination unreachable (type 3) messages, we use the equal to operator in our filter expression:

```
icmp[0] == 3
```

To examine only ICMP packets that represent an echo request (type 8) or echo reply (type 0), use two primitives with the OR operator:

```
icmp[0] == 8 || icmp[0] == 0
```

These filters work great, but they filter based on only 1 byte of information within a packet header. You can also specify the length of the data to be returned in your filter expression by appending the byte length after the offset number within the square brackets, separated by a colon.

For example, say we want to create a filter that captures all ICMP destination-unreachable, host-unreachable packets, identified by type 3, code 1. These are 1-byte fields, located next to each other at offset 0 of the packet header. To do this, we create a filter that checks 2 bytes of data beginning at offset 0 of the packet header, and we compare that data against the hex value 0301 (type 3, code 1), like this:

```
icmp[0:2] == 0x0301
```

A common scenario is to capture only TCP packets with the RST flag set. We will cover TCP extensively in Chapter 8. For now, you just need to know that the flags of a TCP packet are located at offset 13. This is an interesting field because it is collectively 1 byte in size as the flags field, but each particular flag is identified by a single bit within this byte. As I will discuss further in Appendix B, each bit in a byte represents some base 2 number. The bit the flag is stored in is specified by the number the bit represents, so the first bit would represent 1, the second 2, the third 4, and so on. Multiple flags can be set simultaneously in a TCP packet. Therefore, we can't efficiently filter by using a single `tcp[13]` value because several values may represent the RST bit being set.

Instead, we must specify the location within the byte that we wish to examine by appending a single ampersand (`&`), followed by the number that represents where the flag is stored. The RST flag is at the bit representing the number 4 within this byte, and the fact that this bit is set to 4 tells us that the RST flag is set. The filter looks like this:

```
tcp[13] & 4 == 4
```

To see all packets with the PSH flag set, which is identified by the bit location representing the number 8 in the TCP flags at offset 13, our filter would use that location instead:

```
tcp[13] & 8 == 8
```

Sample Capture Filter Expressions

You will often find that the success or failure of your analysis depends on your ability to create filters appropriate for your current situation. Table 4-3 shows a few common capture filters that you might use frequently.

Table 4-3: Commonly Used Capture Filters

Filter	Description
`tcp[13] & 32 == 32`	TCP packets with the URG flag set
`tcp[13] & 16 == 16`	TCP packets with the ACK flag set
`tcp[13] & 8 == 8`	TCP packets with the PSH flag set
`tcp[13] & 4 == 4`	TCP packets with the RST flag set
`tcp[13] & 2 == 2`	TCP packets with the SYN flag set
`tcp[13] & 1 == 1`	TCP packets with the FIN flag set
`tcp[13] == 18`	TCP SYN-ACK packets
`ether host` *`00:00:00:00:00:00`*	Traffic to or from your MAC address
`!ether host` *`00:00:00:00:00:00`*	Traffic not to or from your MAC address
`broadcast`	Broadcast traffic only
`icmp`	ICMP traffic

Filter	Description
`icmp[0:2] == 0x0301`	ICMP destination unreachable, host unreachable
`ip`	IPv4 traffic only
`ip6`	IPv6 traffic only
`udp`	UDP traffic only

Display Filters

A display filter is one that, when applied to a capture file, tells Wireshark to display only packets that match that filter. You can enter a display filter in the Filter text box above the Packet List pane.

Display filters are used more often than capture filters because they allow you to filter the packet data you see without actually omitting the rest of the data in the capture file. That way, if you need to revert to the original capture, you can simply clear the filter expression. They are also a lot more powerful thanks to Wireshark's extensive library of packet dissectors.

As an example, in some situations, you might use a display filter to clear irrelevant broadcast traffic from a capture file by filtering out ARP broadcasts from the Packet List pane when those packets don't relate to the current problem being analyzed. However, because those ARP broadcast packets may be useful later, it's better to filter them temporarily than it is to delete them.

To filter out all ARP packets in the capture window, place your cursor in the Filter text box at the top of the Packet List pane and enter **`!arp`** to remove all ARP packets from the list (Figure 4-16). To remove the filter, click the **X** button, and to save the filter for later, click the plus (+) button.

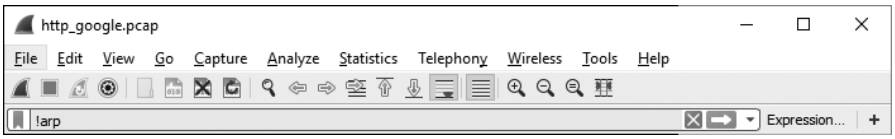


Figure 4-16: Creating a display filter using the Filter text box above the Packet List pane

There are two ways to apply display filters. One is to apply them directly using the appropriate syntax, as we did in this example. Another is to use the Display Filter Expression dialog to build your filter iteratively; this is the easier method when you are first starting to use filters. Let's explore both methods, starting with the easier first.

The Display Filter Expression Dialog

The Display Filter Expression dialog, shown in Figure 4-17, makes it easy for novice Wireshark users to create capture and display filters. To access this dialog, click the **Expression** button on the Filter toolbar.

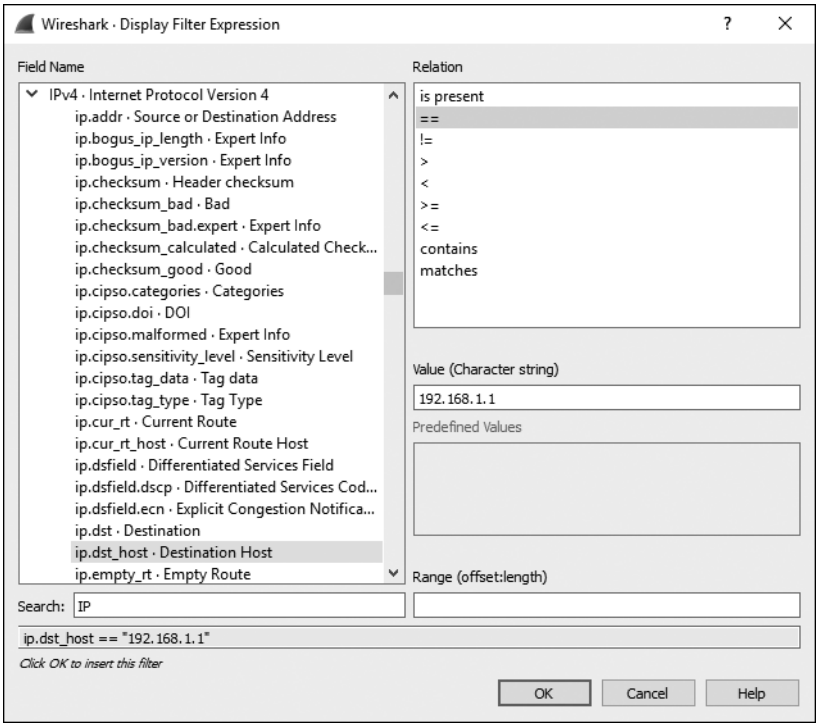


Figure 4-17: The Display Filter Expression dialog allows for the easy creation of filters in Wireshark.

The left side of the dialog lists all possible protocol fields, and these fields specify all possible filter criteria. To create a filter, follow these steps:

1. To view the criteria fields associated with a protocol, expand that protocol by clicking the arrow symbol next to it. Once you find the criterion you want to base your filter on, click to select it.
2. Choose how your selected field will relate to the criterion value you supply. This relation is specified as equal to, greater than, less than, and so on.
3. Create your filter expression by specifying a criterion value that will relate to your selected field. You can define this value or select it from predefined ones programmed into Wireshark.
4. Your complete filter will be displayed at the bottom of the screen. When you've finished, click **OK** to insert it into the filter bar.

The Display Filter Expression dialog is great for novice users, but once you get the hang of things, you'll find that manually entering filter expressions greatly increases your efficiency. The display filter expression syntax structure is simple, yet extremely powerful.

The Filter Expression Syntax Structure

When you begin using Wireshark more, you will want to start using the display filter syntax directly in the main window to save time. Fortunately, the syntax used for display filters follows a standard scheme and is easy to navigate. In most cases, this scheme is protocol-centric and follows the format *protocol.feature.subfeature*, as you saw when looking at the Display Filter Expression dialog. Now we will look at a few examples.

You will most often use a capture or display filter to see packets based on a specific protocol alone. For example, say you are troubleshooting a TCP problem and you want to see only TCP traffic in a capture file. If so, a simple `tcp` filter will do the job.

Now let's look at things from the other side of the fence. Imagine that in the course of troubleshooting your TCP problem, you have used the ping utility quite a bit, thereby generating a lot of ICMP traffic. You could remove this ICMP traffic from your capture file with the filter `expression !icmp`.

Comparison operators allow you to compare values. For example, when troubleshooting TCP/IP networks, you will often need to view all packets that reference a particular IP address. The equal to comparison operator (`==`) will allow you to create a filter showing all packets with an IP address of 192.168.0.1:

```
ip.addr==192.168.0.1
```

Now suppose that you need to view only packets that are less than 128 bytes. You can use the less than or equal to operator (`<=`) to accomplish this goal:

```
frame.len<=128
```

Table 4-4 shows Wireshark's comparison operators.

Table 4-4: Wireshark Filter Expression Comparison Operators

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Logical operators allow you to combine multiple filter expressions into one statement, dramatically increasing the effectiveness of your filters.

For example, say that you're interested in displaying only packets to two IP addresses. You can use the `or` operator to create one expression that will display packets containing either IP address, like this:

```
ip.addr==192.168.0.1 or ip.addr==192.168.0.2
```

Table 4-5 lists Wireshark's logical operators.

Table 4-5: Wireshark Filter Expression Logical Operators

Operator	Description
`and`	Both conditions must be true.
`or`	Either one of the conditions must be true.
`xor`	One and only one condition must be true.
`not`	Neither one of the conditions is true.

Sample Display Filter Expressions

Although the concepts related to creating filter expressions are fairly simple, you will need to use several specific keywords and operators when creating new filters for various problems. Table 4-6 shows some of the display filters that I use most often. For a complete list, see the Wireshark display filter reference at *http://www.wireshark.org/docs/dfref/*.

Table 4-6: Commonly Used Display Filters

Filter	Description
`!tcp.port==3389`	Filter out RDP traffic
`tcp.flags.syn==1`	TCP packets with the SYN flag set
`tcp.flags.reset==1`	TCP packets with the RST flag set
`!arp`	Clear ARP traffic
`http`	All HTTP traffic
`tcp.port==23 || tcp.port==21`	Telnet or FTP traffic
`smtp || pop || imap`	Email traffic (SMTP, POP, or IMAP)

Saving Filters

Once you begin creating a lot of capture and display filters, you will find that you use certain ones frequently. Fortunately, you don't need to type these in each time you want to use them, because Wireshark lets you save your filters for later use. To save a custom capture filter, follow these steps:

1. Select **Capture ▸ Capture Filters** to open the Capture Filter dialog.
2. Create a new filter by clicking the plus (+) button on the lower left side of the dialog.

3. Enter a name for your filter in the Filter Name box.
4. Enter the actual filter expression in the Filter String box.
5. Click the **OK** button to save your filter expression in the list.

To save a custom display filter, follow these steps:

1. Type your filter into the Filter bar above the Packet List pane in the main window and click the **ribbon** button on the left side of the bar.
2. Click the **Save this Filter** option, and a list of saved display filters will be presented in a separate dialog. There you can provide a name for your filter before clicking **OK** to save it (Figure 4-18).

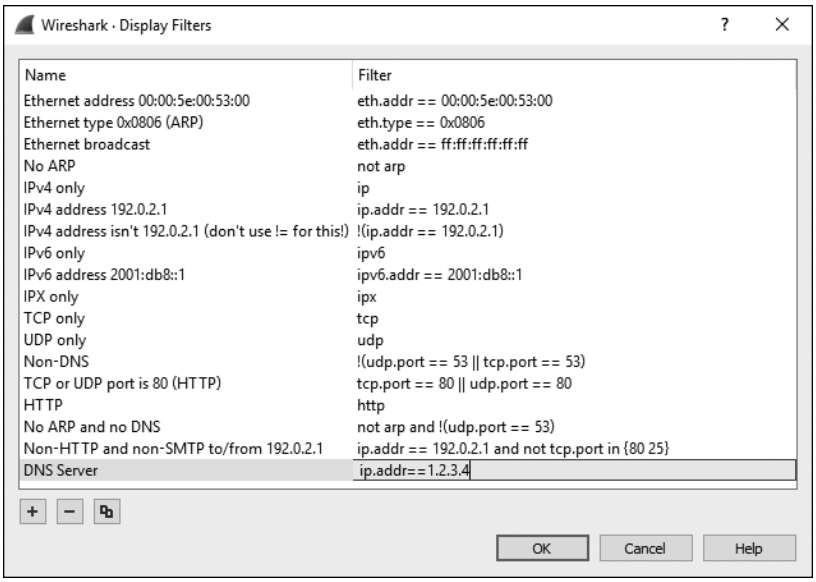


Figure 4-18: You can save display filters directly from the main toolbar.

Adding Display Filters to a Toolbar

If you have filters that you find yourself flipping on and off frequently, one of the easiest ways to interact with them is to add filter toggles to the Filter bar just above the Packet List pane. To do this, complete the following steps:

1. Type your filter into the Filter bar above the Packet List pane in the main window and click the plus (+) button on the right side of the bar.
2. A new bar will display below the Filter bar where you can provide a name for your filter in the Label field (Figure 4-19). This is the label that will be used to represent the filter on the toolbar. Once you've input something in this field, click **OK** to create a shortcut to this expression in the Filter toolbar.

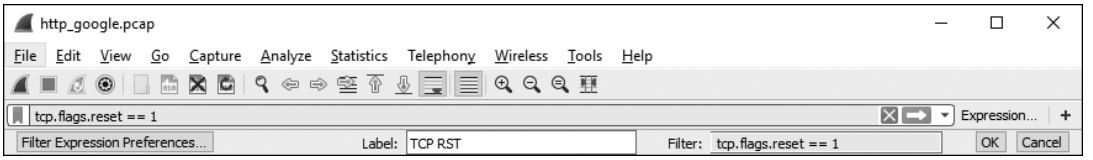


Figure 4-19: Adding a filter expression shortcut to the Filter toolbar

As you can see in Figure 4-20, we've created a shortcut to a filter that will quickly show any TCP packets with the RST flag enabled. Additions to the filtering toolbar are saved to your configuration profile (as discussed in Chapter 3), making them a powerful way to enhance your ability to identify problems in packet captures in various scenarios.

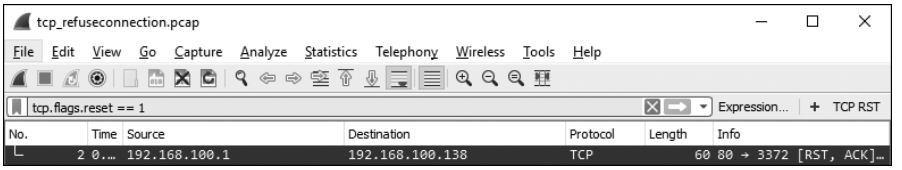


Figure 4-20: Filtering using a toolbar shortcut

Wireshark includes several built-in filters that are great examples of what a filter should look like. You'll want to use them (together with the Wireshark help pages) when creating your own filters. We'll use filters in examples throughout this book.