

3

INTRODUCTION TO WIRESHARK



As mentioned in Chapter 1, several packet-sniffing applications are available for performing network analysis, but we'll use Wireshark in this book. This chapter introduces Wireshark.

A Brief History of Wireshark

Wireshark has a very rich history. Gerald Combs, a computer science graduate of the University of Missouri at Kansas City, originally developed it out of necessity. The first version of Combs's application, called Ethereal, was released in 1998 under the GNU Public License (GPL).

Eight years after releasing Ethereal, Combs left his job to pursue other career opportunities. Unfortunately, his employer at that time had full rights to the Ethereal trademarks, and Combs was unable to reach an agreement that would allow him to control the Ethereal "brand." Instead, Combs and the rest of the development team rebranded the project as *Wireshark* in mid-2006.

Wireshark has grown dramatically in popularity, and its collaborative development team now boasts more than 500 contributors. The program as it exists under the Ethereal name is no longer being developed.

The Benefits of Wireshark

Wireshark offers several benefits that make it appealing for everyday use. It is aimed at both the journeyman and the expert packet analyst, and offers a variety of features to entice each. Let's examine Wireshark according to the criteria defined in Chapter 1 for selecting a packet-sniffing tool.

Supported protocols Wireshark excels in the number of protocols that it supports—more than 850 as of this writing. These range from common ones like IP and DHCP to more advanced proprietary protocols like AppleTalk and BitTorrent. And because Wireshark is developed under an open source model, new protocol support is added with each update.

NOTE *In the unlikely case that Wireshark doesn't support a protocol you need, you can code that support yourself and submit your code to the Wireshark developers for inclusion in the application (if your code is accepted, of course).*

User-friendliness The Wireshark interface is one of the easiest to understand of any packet-sniffing application. It is GUI-based, with very clearly written context menus and a straightforward layout. It also provides several features designed to enhance usability, such as protocol-based color coding and detailed graphical representations of raw data. Unlike some of the more complicated command-line-driven alternatives, like tcpdump, the Wireshark GUI is great for those who are just entering the world of packet analysis.

Cost Since it is open source, Wireshark's pricing can't be beat: Wireshark is released as free software under the GPL. You can download and use Wireshark for any purpose, whether personal or commercial.

NOTE *Although Wireshark may be free, some people have made the mistake of paying for it by accident. If you search for packet sniffers on eBay, you may be surprised by how many people would love to sell you a "professional enterprise license" for Wireshark for the low, low price of \$39.95. Of course, this is a farce, but if you decide you really want to buy it, give me a call, and we can talk about some oceanfront property in Kentucky I have for sale!*

Program support A software package's level of support can make or break it. When dealing with freely distributed software such as Wireshark, there may not be any formal support, which is why the open source community often relies on its user base to provide support. Luckily for us, the Wireshark community is one of the most active of any open source project.

The Wireshark web page links directly to several forms of support, including online documentation, a support and development wiki, FAQs, and a place to sign up for the Wireshark mailing list, which is monitored by most of the program's top developers. Paid support for Wireshark is also available from CACE Technologies through its SharkNet program.

Operating system support Wireshark supports all major modern operating systems, including Windows, Mac OS X, and Linux-based platforms. You can view a complete list of supported operating systems on the Wireshark home page.

Installing Wireshark

The Wireshark installation process is surprisingly simple. However, before you install Wireshark, make sure that your system meets the following requirements:

- 400 MHz processor or faster
- 128MB RAM
- At least 75MB of available storage space
- NIC that supports promiscuous mode
- WinPcap capture driver

The WinPcap capture driver is the Windows implementation of the pcap packet-capturing application programming interface (API). Simply put, this driver interacts with your operating system to capture raw packet data, apply filters, and switch the NIC in and out of promiscuous mode.

Although you can download WinPcap separately (from <http://www.winpcap.org/>), it is typically better to install WinPcap from the Wireshark installation package, because the included version of WinPcap has been tested to work with Wireshark.

Installing on Microsoft Windows Systems

The first step when installing Wireshark under Windows is to obtain the latest installation build from the official Wireshark web page, <http://www.wireshark.org/>. Navigate to the Downloads section on the website and choose a mirror. Once you've downloaded the package, follow these steps:

1. Double-click the *.exe* file to begin installation, and then click **Next** in the introductory window.
2. Read the licensing agreement, and then click **I Agree** if you agree.
3. Select the components of Wireshark you wish to install, as shown in Figure 3-1. For our purposes, you can accept the defaults by clicking **Next**.

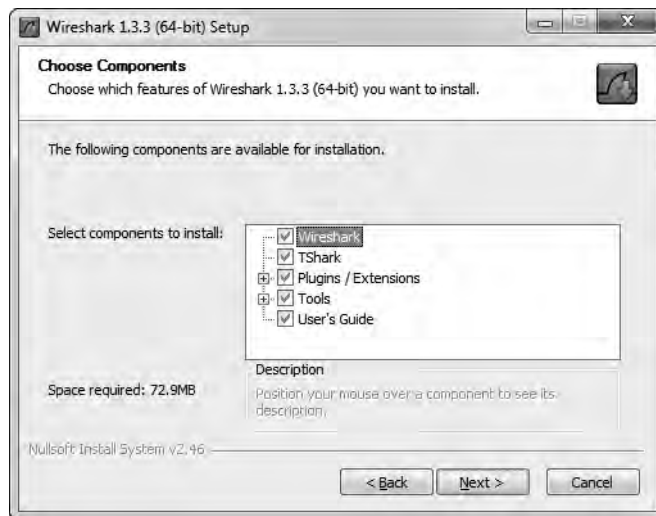


Figure 3-1: Choosing the Wireshark components you wish to install

4. Click **Next** in the Additional Tasks window.
5. Select the location where you wish to install Wireshark, and then click **Next**.
6. When the dialog asks whether you want to install WinPcap, make sure the **Install WinPcap** box is checked, as shown in Figure 3-2, and then click **Install**. The installation process should begin.

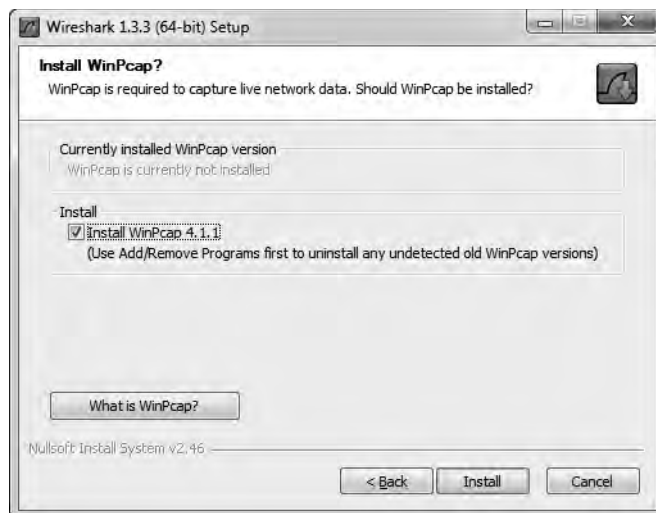


Figure 3-2: Selecting the option to install the WinPcap driver

7. About halfway through the Wireshark installation, the WinPcap installation should start. When it does, click **Next** in the introductory window, read the licensing agreement, and then click **I Agree**.

8. WinPcap should install on your computer. After this installation is complete, click **Finish**.
9. Wireshark should complete its installation. When it's finished, click **Next**.
10. In the installation confirmation window, click **Finish**.

Installing on Linux Systems

The first step when installing Wireshark on a Linux system is to download the appropriate installation package. Not every version of Linux is supported, so don't be surprised if your specific distribution doesn't have its own install package.

Typically, for system-wide software, root access is a requirement. However, local software installations compiled from source can usually be installed without root access.

RPM-based Systems

For RPM-based distributions, such as Red Hat Linux, download the appropriate installation package from the Wireshark web page. Then open a console window and enter the following (substituting the filename of your downloaded package as appropriate):

```
rpm -ivh wireshark-0.99.3.i386.rpm
```

If any dependencies are missing, install them and repeat the Wireshark installation.

DEB-based Systems

On a DEB-based distribution such as Debian or Ubuntu, you can install Wireshark from the system repositories. Open a console window and type the following:

```
apt-get install wireshark
```

Compiling from Source

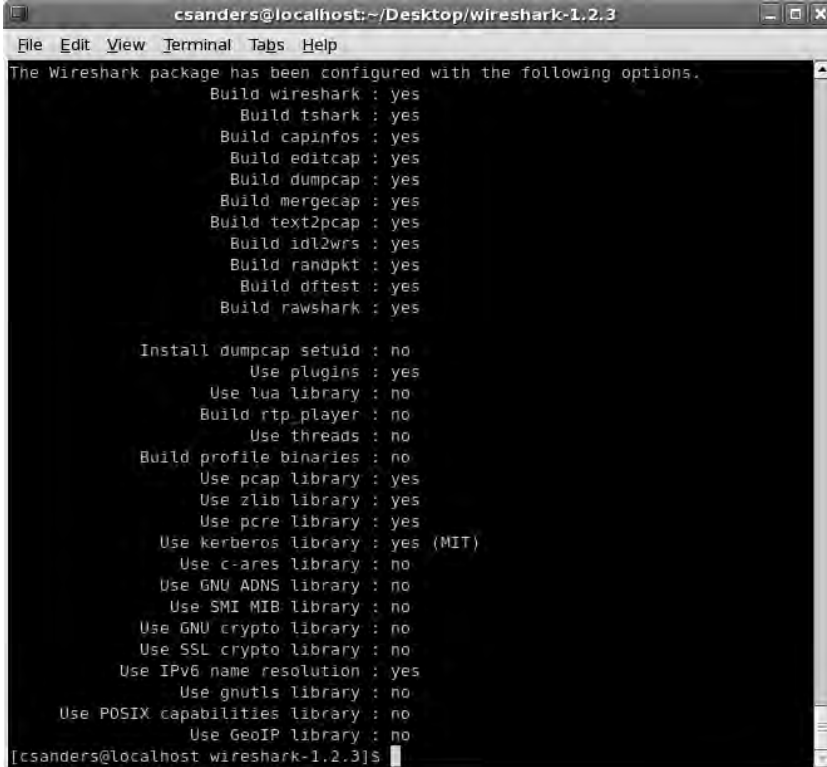
If your Linux distribution doesn't use an automated package management software, the most effective way to install Wireshark is to compile it from source. To do this, complete the following steps:

1. Download the source package from the Wireshark web page.
2. Extract the archive by typing the following (substituting the filename of your downloaded package as appropriate):

```
tar -jxvf wireshark-1.2.2.tar.bz2
```

3. Change into the newly created directory where the files were extracted.

4. As a root-level user, configure the source so that it will build correctly for your distribution of Linux by using the command `./configure`. If you wish to deviate from the default installation options, you can specify those options at this point in the installation. If any dependencies are missing, you will most likely receive an error. If installation is successful, you should see a message noting success, as shown in Figure 3-3.



```
csanders@localhost:~/Desktop/wireshark-1.2.3
File Edit View Terminal Tabs Help
The Wireshark package has been configured with the following options.
Build wireshark : yes
Build tshark : yes
Build capinfos : yes
Build editcap : yes
Build dumpcap : yes
Build mergecap : yes
Build text2pcap : yes
Build idl2wrs : yes
Build randpkt : yes
Build oftest : yes
Build rawshark : yes

Install dumpcap setuid : no
Use plugins : yes
Use lua library : no
Build rtp_player : no
Use threads : no
Build profile binaries : no
Use pcap library : yes
Use zlib library : yes
Use pcre library : yes
Use kerberos library : yes (MIT)
Use c-ares library : no
Use GNU ADNS library : no
Use SMI MIB library : no
Use GNU crypto library : no
Use SSL crypto library : no
Use IPv6 name resolution : yes
Use gnutls library : no
Use POSIX capabilities library : no
Use GeoIP library : no
[csanders@localhost wireshark-1.2.3]$
```

Figure 3-3: Successful output from the `./configure` command

5. Enter the `make` command to build the source into a binary.
6. Initiate the final installation with `make install`.

Installing on Mac OS X Systems

There are a few caveats for installing Wireshark on Mac OS X Snow Leopard, but installation is not a difficult task and I've outlined the installation steps here. The steps are:

1. Download the DMG package from the Wireshark web page.
2. Copy *Wireshark.app* to the *Applications* folder.
3. Open the *Utilities* folder in *Wireshark.app*.

4. In Finder, click **Go**, and select **Go To Folder**. Enter `/usr/local/bin/` to open that directory.
5. Copy the contents of the *Command Line* folder into `/usr/local/bin/`. You must enter your password in order to do this.
6. In the *Utilities* folder, copy the *ChmodBPF* folder into the *StartupItems* folder. You will need to enter your password again to perform this action and complete the installation.

Wireshark Fundamentals

Once you've successfully installed Wireshark on your system, you can begin to familiarize yourself with it. Now you finally get to open your fully functioning packet sniffer and see . . . absolutely nothing!

Okay, so Wireshark isn't very interesting when you first open it. In order for things to really get exciting, you need to get some data.

Your First Packet Capture

To get packet data into Wireshark, you'll perform your first packet capture. You may be thinking, "How am I going to capture packets when nothing is wrong on the network?"

First, there is *always* something wrong on the network. If you don't believe me, then go ahead and send an email to all of your network users and let them know that everything is working perfectly.

Secondly, there doesn't need to be something wrong in order for you to perform packet analysis. In fact, most packet analysts spend more time analyzing problem-free traffic than traffic that they are troubleshooting. You need a baseline to compare to in order to be able to effectively troubleshoot network traffic. For example, if you ever hope to solve a problem with DHCP by analyzing its traffic, you must understand what the flow of working DHCP traffic looks like.

More broadly, in order to find anomalies in daily network activity, you must know what normal daily network activity looks like. When your network is running smoothly, you can set your baseline so that you'll know what its traffic looks like in a normal state.

So, let's capture some packets!

1. Open Wireshark.
2. From the main drop-down menu, select **Capture** and then **Interfaces**. You should see a dialog listing the various interfaces that can be used to capture packets, along with their IP addresses.
3. Choose the interface you wish to use, as shown in Figure 3-4, and click **Start**, or simply click the interface under the Interface List section of the welcome page. Data should begin filling the window.



Figure 3-4: Selecting an interface on which to perform your packet capture

- Wait about a minute or so, and when you are ready to stop the capture and view your data, click the **Stop** button from the Capture drop-down menu.

Once you have completed these steps and finished the capture process, the Wireshark main window should be alive with data. As a matter of fact, you might be overwhelmed by the amount of data that appears, but it will all start to make sense very quickly as we break down the main window of Wireshark one piece at a time.

Wireshark's Main Window

You'll spend most of your time in the Wireshark main window. This is where all of the packets you capture are displayed and broken down into a more understandable format. Using the packet capture you just made, let's take a look at Wireshark's main window, as shown in Figure 3-5.

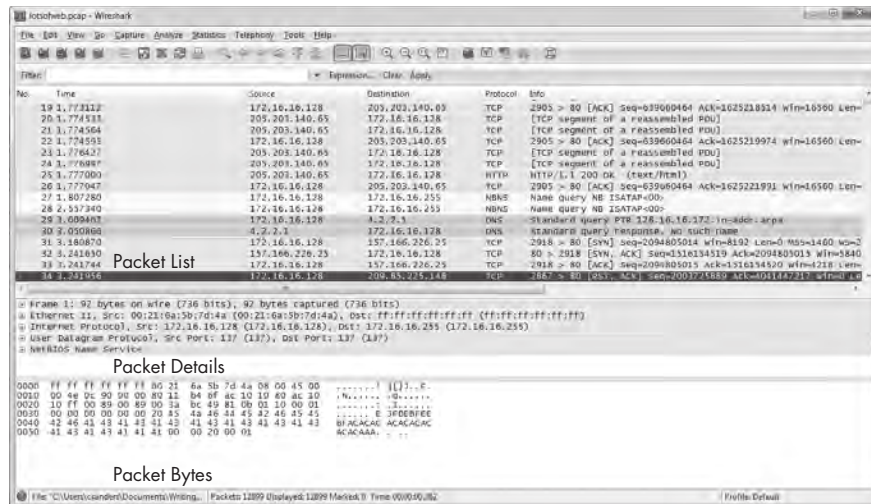


Figure 3-5: The Wireshark main window uses a three-pane design.

The three panes in the main window depend on one another. In order to view the details of an individual packet in the Packet Details pane, you must first select that packet by clicking it in the Packet List pane. Once you've selected your packet, you can see the bytes that correspond with a certain portion of the packet in the Packet Bytes pane when you click that portion of the packet in the Packet Details pane.

NOTE *Notice that Figure 3-5 lists a few different protocols in the Packet List pane. There is no visual separation of protocols on different layers; all packets are shown as they are received on the wire.*

Here's what each pane contains:

Packet List The top pane displays a table containing all packets in the current capture file. It has columns containing the packet number, the relative time the packet was captured, the source and destination of the packet, the packet's protocol, and some general information found in the packet.

NOTE *When I refer to traffic, I am referring to all packets displayed in the Packet List pane. When I refer to DNS traffic specifically, I mean the DNS protocol packets in the Packet List pane.*

Packet Details The middle pane contains a hierarchical display of information about a single packet. This display can be collapsed and expanded to show all of the information collected about an individual packet.

Packet Bytes The lower pane—perhaps the most confusing—displays a packet in its raw, unprocessed form; that is, it shows what the packet looks like as it travels across the wire. This is raw information with nothing warm or fuzzy to make it easier to follow.

Wireshark Preferences

Wireshark has several preferences that can be customized to meet your needs. To access Wireshark's preferences, select **Edit** from the main drop-down menu and click **Preferences**. You'll see the Preferences dialog, which contains several customizable options, as shown in Figure 3-6.

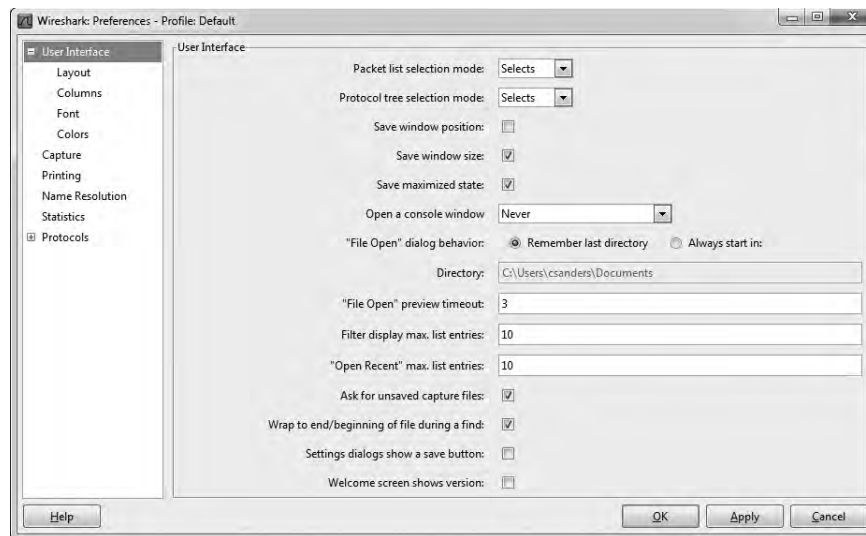


Figure 3-6: You can customize Wireshark using the Preferences dialog options.

Wireshark's preferences are divided into six major sections:

User Interface These preferences determine how Wireshark presents data. You can change most options here according to your personal preferences, including whether or not to save window positions, the layout of the three main panes, the placement of the scroll bar, the placement of the Packet List pane columns, the fonts used to display the captured data, and the background and foreground colors.

Capture These preferences allow you to specify options related to the way packets are captured, including your default capture interface, whether to use promiscuous mode by default, and whether to update the Packet List pane in real time.

Printing The preferences in this section allow you to specify various options related to the way Wireshark prints your data.

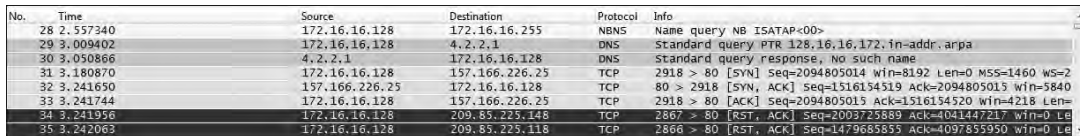
Name Resolution Through these preferences, you can activate features of Wireshark that allow it to resolve addresses into more recognizable names (including MAC, network, and transport name resolution) and specify the maximum number of concurrent name resolution requests.

Statistics This section provides a few configurable options for Wireshark's statistical features.

Protocols The preferences in this section allow you to manipulate options related to the capture and display of the various packets Wireshark is capable of decoding. Not every protocol has configurable preferences, but some have several options that can be changed. These options are best left at their defaults unless you have a specific reason to change them.

Packet Color Coding

If you are anything like me, you may enjoy shiny objects and pretty colors. If that is the case, you probably got excited when you saw all those different colors in the Packet List pane, as in the example in Figure 3-7 (well, the figure is in black and white, but you get the idea). It may seem as if these colors are randomly assigned to each individual packet, but this is not the case.



No.	Time	Source	Destination	Protocol	Info
28	2.557340	172.16.16.128	172.16.16.255	NBNS	Name query NB ISATAP<00>
29	3.009402	172.16.16.128	4.2.2.1	DNS	Standard query PTR 128.16.16.172.in-addr.arpa
30	3.050866	4.2.2.1	172.16.16.128	DNS	Standard query response, No such name
31	3.180870	172.16.16.128	157.166.226.25	TCP	2918 > 80 [SYN] seq=2094805014 win=8192 Len=0 MSS=1460 WS=2
32	3.241650	157.166.226.25	172.16.16.128	TCP	80 > 2918 [SYN, ACK] seq=1516154519 Ack=2094805015 win=5840
33	3.241744	172.16.16.128	157.166.226.25	TCP	2918 > 80 [ACK] seq=2094805015 Ack=1516154520 win=4218 Len=
34	3.241956	172.16.16.128	209.85.225.148	TCP	2867 > 80 [RST, ACK] Seq=2003725859 Ack=4041447217 Win=0 Len=
35	3.242062	172.16.16.128	209.85.225.118	TCP	2866 > 80 [RST, ACK] Seq=1479688835 Ack=4037855950 Win=0 Len=

Figure 3-7: Wireshark's color coding allows for quick protocol identification.

Each packet is displayed as a certain color for a reason. These colors reflect the packet's protocol. For example, all DNS traffic is blue, and all HTTP traffic is green. The color coding allows you to quickly differentiate between various protocols so that you don't need to read the protocol field in the Packet List pane for each individual packet. You will find that this greatly speeds up the time it takes to browse through large capture files.

Wireshark makes it easy to see which colors are assigned to each protocol through the Coloring Rules window, shown in Figure 3-8. To open this window, select **View** from the main drop-down menu and click **Coloring Rules**.

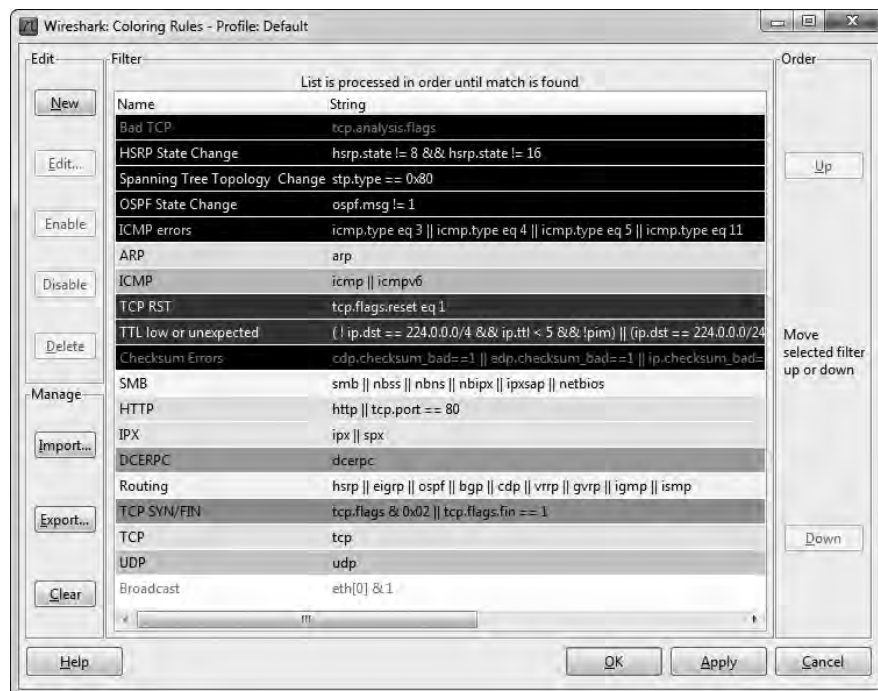


Figure 3-8: The Coloring Rules window allows you to view and modify the coloring of packets.

You can define your own coloring rules and modify existing ones. For example, to change the color used as the background for HTTP traffic from the default green to lavender, follow these steps:

1. Open Wireshark and access the Coloring Rules window (**View ▶ Coloring Rules**).
2. Find the HTTP coloring rule in the coloring rules list and select it by clicking it once.
3. Click the **Edit** button. You'll see the Edit Color Filter dialog, as shown in Figure 3-9.

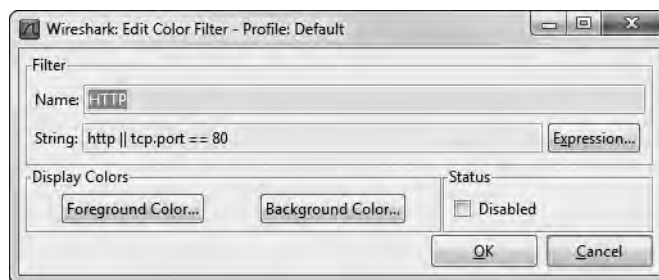


Figure 3-9: When editing a color filter, you can modify both the foreground and background colors.

4. Click the **Background Color** button.
5. Select the color you wish to use on the color wheel, and then click **OK**.
6. Click **OK** twice more to accept the changes and return to the main window. The main window should then reload itself to reflect the updated color scheme.

As you work with Wireshark on your network, you will begin to notice that you deal with certain protocols more than others. Here's where color-coded packets can make your life a lot easier. For example, if you think that there is a rogue DHCP server on your network handing out IP leases, you could simply modify the coloring rule for the DHCP protocol so that it shows up in bright yellow (or some other easily identifiable color). This would allow you to pick out all DHCP traffic much more quickly, and make your packet analysis more efficient.

These coloring rules can also be further extended by creating them based on your own custom filters.

Now that you have Wireshark up and running, you're ready to do some packet analysis. The next chapter describes how you can work with the packets you've captured.

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

As you perform packet analysis, you will find that a good portion of the analysis you do will happen after your capture. Usually, you will 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're asked for a location to save your packet capture and for the file format you wish to use. If you do not specify a file format, Wireshark will use the default *.pcap* file format.

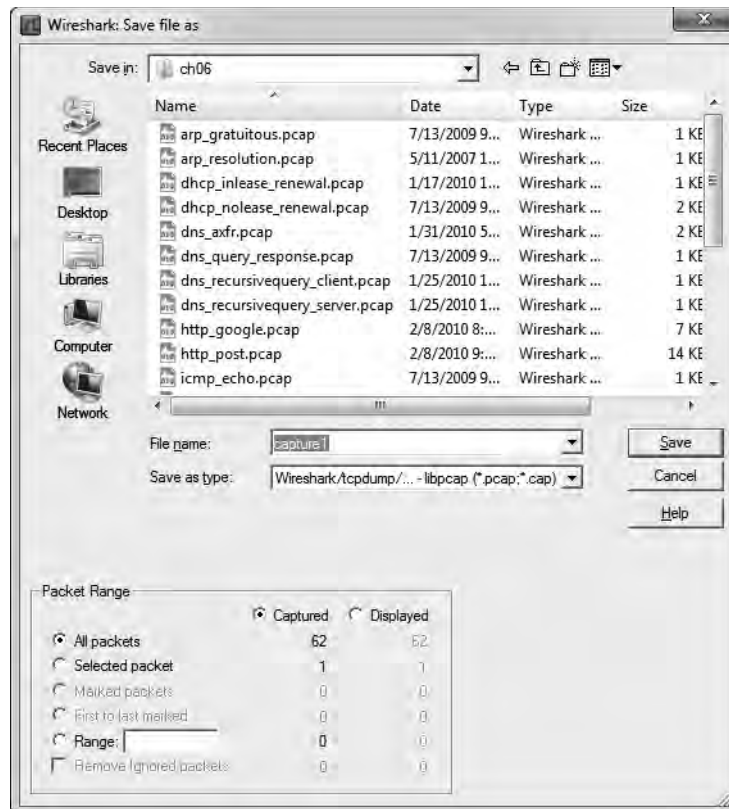


Figure 4-1: The Save File As dialog allows you to save your packet captures.

One of the more powerful features of the Save File As dialog is the ability to save a specific packet range. 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 different 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, choose **File ▶ Export**, and then select the format for the exported file. You will see a Save As dialog containing options related to that specific format.

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 capture files you want to merge and choose **File ▶ Merge** to bring up the Merge with Capture File dialog, shown in Figure 4-2. 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.

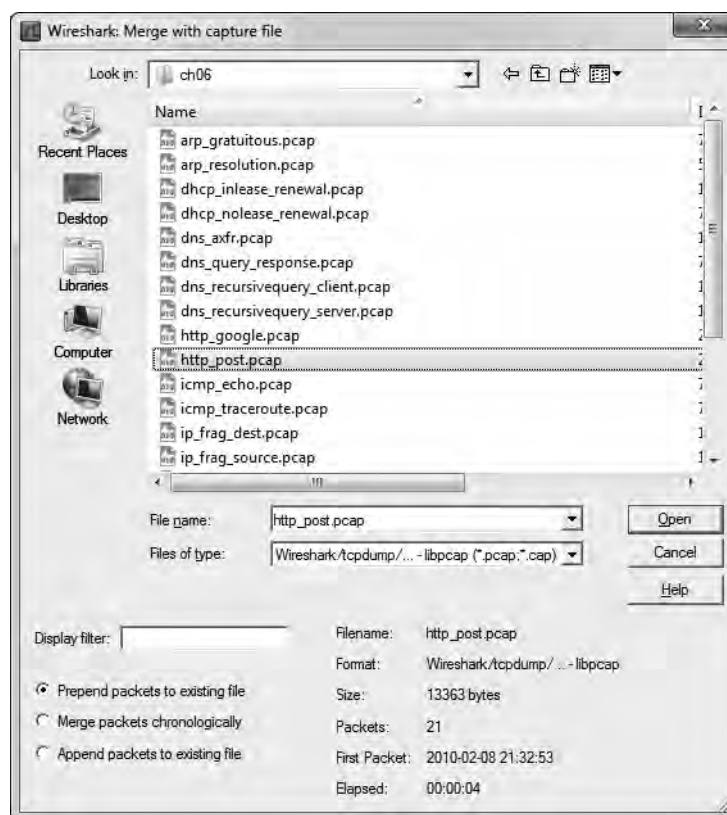


Figure 4-2: The Merge with Capture File dialog allows you to merge two capture files.

Working with Packets

You will eventually encounter situations involving a very large number of packets. As the number of these packets grows into the thousands and even millions, you will need to be able 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 dialog, shown in Figure 4-3, by pressing CTRL-F.

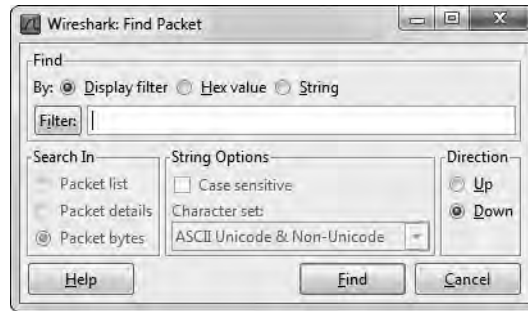


Figure 4-3: Finding packets in Wireshark based on specified criteria

This dialog 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.
- The Hex value option searches for packets with a hexadecimal (with bytes separated by colons) value you specify.
- The String option searches for packets with a text string you specify.

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	00:ff ff:ff 00:AB:B1:f0
String	Workstation1 UserB domain

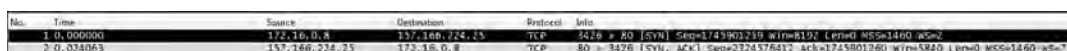
Other options include the ability to select the window in which you want to search, the character set to use, and the search direction. You can extend the capability of your string searches by specifying the pane the search is performed in, setting the character set used, and making the search case sensitive.

Once you've made your selections, enter your search criteria in the text box, and click **Find** to find the first packet that meets your criteria. To find the next matching packet, press CTRL-N; find the previous matching packet by pressing CTRL-B.

Marking Packets

After you have found the packets that match your criteria, you can mark those of particular interest. For example, you may want to mark packets to be able to save those packets separately or to find them quickly based on the coloration. Marked packets stand out with a black background and white text, as shown in Figure 4-4. (You can also sort out only marked packets when saving packet captures.)

To mark a packet, 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 using 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.



No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.0.8	157.140.224.25	TCP	4226 → 80 [SYN] Seq=1743903219 win=8192 Len=0 MSS=1460 WSeq=
2	0.034063	157.140.224.25	172.16.0.8	TCP	80 → 3426 [SYN, ACK] Seq=2324576412 Ack=1743901260 Win=5840 Len=0 MSS=1460 WSeq=

Figure 4-4: A marked packet is highlighted on your screen. In this example, packet 1 is marked and appears darker.

Printing Packets

Although most analysis will take place on the computer screen, you may need to print captured data. I often print out packets and tape them to my desk so that 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. You will see the Print dialog, as shown in Figure 4-5.

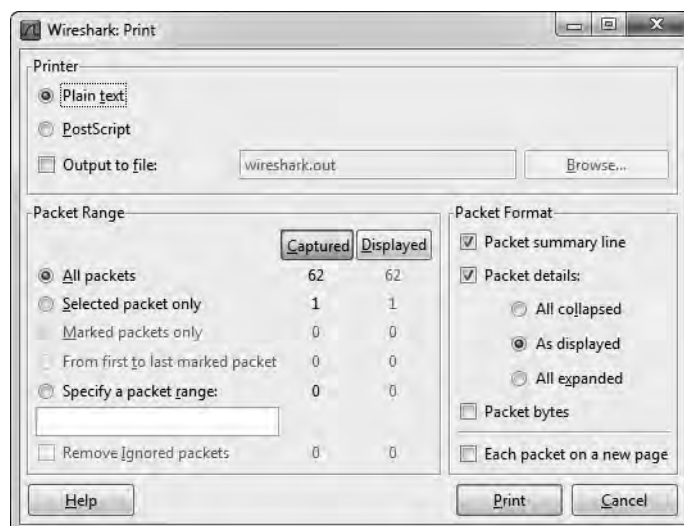


Figure 4-5: The Print dialog allows you to print the packets you specify.

You can print the selected data as plaintext or PostScript, or to an output file. As with the Save File As dialog, you can print a specific packet range, marked packets only, or packets displayed as the result of a filter. You can also select which of Wireshark's three main panes 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 nearly every capture file. Wireshark recognizes the importance of time and supplies several configurable options relating to it. 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 indicating 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.

The options related to the time display are found under the View heading on the main menu. The Time Display Format section, shown in Figure 4-6, lets you configure the presentation format as well as the precision of the time display. The presentation format option lets you choose various options for time display. The precision options allow you to set the time display precision to automatic 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.

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 specific packet. This feature is particularly handy when you are examining a series of sequential events that are triggered somewhere other than the start of the capture file.

To set a time reference to a certain packet, select the reference packet in the Packet List pane, and then choose **Edit ▶ Set Time Reference** from the main menu. To remove a time reference from a certain packet, select the packet and toggle off the **Edit ▶ Set Time Reference** setting.

When you enable a time reference on a particular packet, the Time column in the Packet List pane will display *REF*, as shown in Figure 4-7.

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 will create a set of times that can be very confusing.

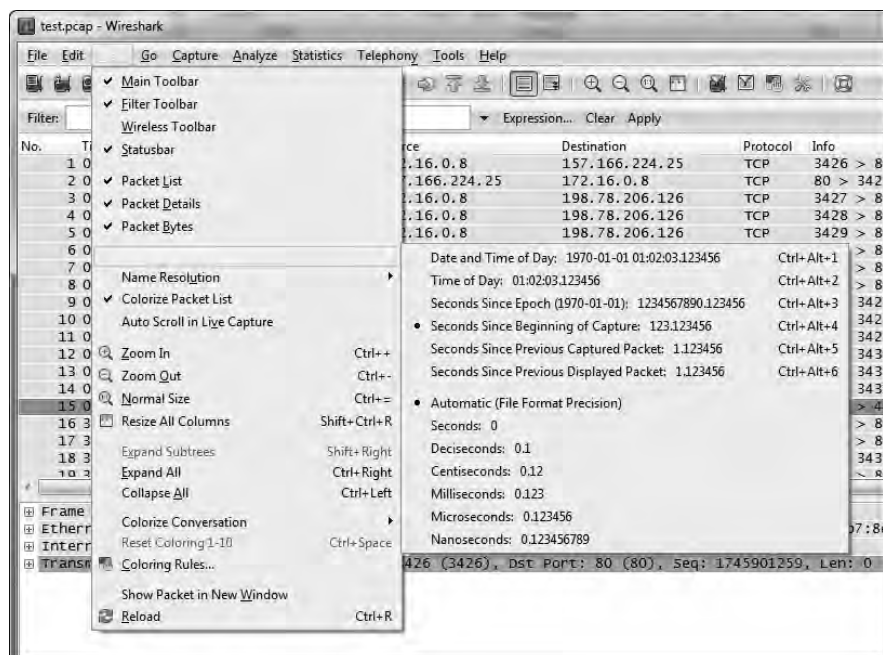


Figure 4-6: Several time display formats are available.

No.	Time	Source	Destination
4	0.118129	172.16.0.8	198.78.206.126
5	* REF *	172.16.0.8	198.78.206.126
6	0.000077	172.16.0.8	198.78.206.126
7	0.000153	172.16.0.8	198.78.206.126

Figure 4-7: A packet with the packet time reference toggle enabled

Setting Capture Options

We walked through a very basic packet capture in Chapter 3. Wireshark offers quite a few more capture options in the Capture Options dialog, shown in Figure 4-8. To open this dialog, choose **Capture** ► **Interfaces** and click the **Options** button next to the interface on which you want to capture packets.

The Capture Options dialog has more bells and whistles than you can shake a stick at, all designed to give you more flexibility while capturing packets. It's divided into Capture, Capture Files, Stop Capture, Display Options, and Name Resolution sections, which we'll examine separately.

Capture Settings

The Interface drop-down list in the Capture section is where you can select the network interface to configure. The left drop-down list allows you to specify whether the interface is local or remote, and the right drop-down list shows all available capture interfaces. The IP address of the interface you have selected is displayed directly below this drop-down list.

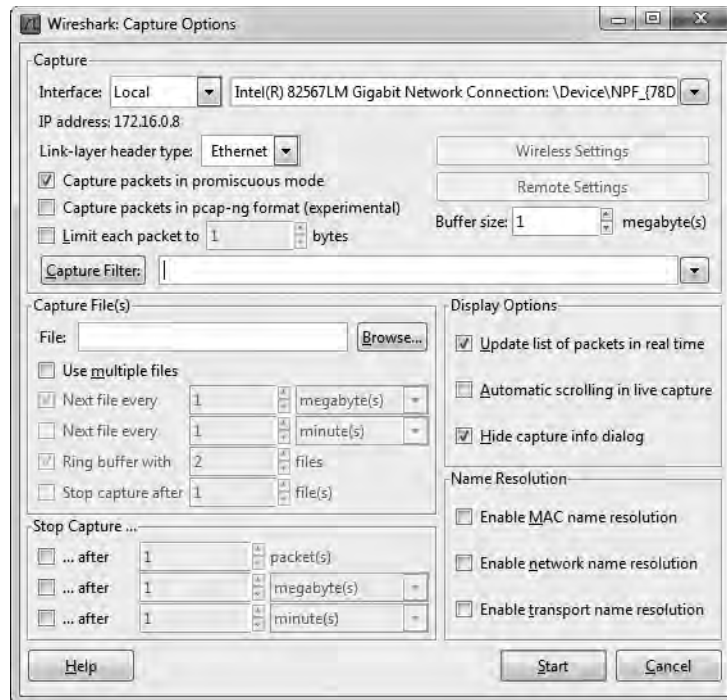


Figure 4-8: The Capture Options dialog

The three checkboxes on the left side of the dialog box allow you to enable or disable promiscuous mode (always enabled by default), capture packets in the currently experimental pcap-ng format, and limit the size of each capture packet by bytes.

The buttons on the right side of the Capture section let you access wireless and remote settings (as applicable). Beneath those is the buffer size option, which is available only on systems running Microsoft Windows. You can specify the amount of capture packet data that is stored in the kernel buffer before it is written to disk. (This is a value you won't normally modify unless you begin noticing that you are dropping a lot of packets.) The Capture Filter option lets you specify a capture filter.

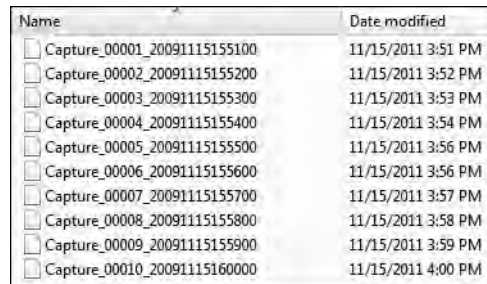
Capture File(s) Settings

The Capture File(s) section allows you to automatically store capture packets in a file, rather than capturing them first and then saving the file. Doing so offers you a great deal 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 to manage the number of files created. To enable this option, enter a complete file path and name in the File text box.

When 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 Use Multiple Files option here.

Wireshark uses various triggers to manage saving to file sets based upon a file size or time condition. To enable these options, place a check mark next to the Next File Every option (the top one for file-size triggers and the one beneath that for time-based triggers), and then specify the value and unit on which to trigger. For instance, you can create a trigger that creates a new file after every 1MB of traffic captured, or after every minute of traffic captured, as shown in Figure 4-9.



Name	Date modified
<input type="checkbox"/> Capture_00001_20091115155100	11/15/2011 3:51 PM
<input type="checkbox"/> Capture_00002_20091115155200	11/15/2011 3:52 PM
<input type="checkbox"/> Capture_00003_20091115155300	11/15/2011 3:53 PM
<input type="checkbox"/> Capture_00004_20091115155400	11/15/2011 3:54 PM
<input type="checkbox"/> Capture_00005_20091115155500	11/15/2011 3:56 PM
<input type="checkbox"/> Capture_00006_20091115155600	11/15/2011 3:56 PM
<input type="checkbox"/> Capture_00007_20091115155700	11/15/2011 3:57 PM
<input type="checkbox"/> Capture_00008_20091115155800	11/15/2011 3:58 PM
<input type="checkbox"/> Capture_00009_20091115155900	11/15/2011 3:59 PM
<input type="checkbox"/> Capture_00010_20091115160000	11/15/2011 4:00 PM

Figure 4-9: A file set created by Wireshark at one-minute intervals

These options can also be used in combination. For example, if you specify both triggers, a new file will be created when 1MB of data is captured *or* when a minute has elapsed—whichever comes first.

The Ring Buffer With option lets you use a ring buffer when creating a file set. This is used by Wireshark as a first in, first out (FIFO) method of writing multiple files. Although the term *ring buffer* has multiple meanings throughout information technology, for our purposes here, it is essentially a file set that specifies that upon completion of writing the last file, the first file is overwritten when more data must be saved to disk. 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.

The Stop Capture After option allows you to stop the current capture once a certain number of files have been created.

Stop Capture Settings

The Stop Capture section lets you stop the running capture after certain triggers are met. As with multiple file sets, you can trigger based on file size and time interval, as well as number of packets. These options can be used with the multiple file options previously discussed.

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 Automatic Scrolling in 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 Automatic Scrolling in Live Capture options can be quite processor intensive, even when capturing a reasonable amount of data. Unless you have a specific need to see the packets in real time, it's best to deselect both options.*

The Hide Capture Info Dialog option lets you suppress the display of a small window that shows the number and percentage of packets that have been captured, by protocol.

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 in Wireshark more in depth, including its drawbacks, in Chapter 5.

Using Filters

Filters allow you to specify exactly 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 used during the actual packet-capturing process. 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 dealing with large amounts of data. The analysis process can be sped up by ensuring that you are looking at only the packet relevant to the issue at hand.

A simple example of when you might use a capture filter is when capturing traffic on a server with multiple roles. Suppose you are troubleshooting an issue with a service running on port 262. If the server you are analyzing runs several different services on a variety of ports, finding and analyzing only the traffic on port 262 can be quite a job in itself. To capture only the port 262 traffic, you can use a capture filter. To do so, you can use the Capture Options dialog, discussed earlier in this chapter, as follows:

1. Choose **Capture ▶ Interfaces** and click the **Options** button next to the interface on which you want to capture packets to open the Capture Options dialog.
2. Select the interface you wish to capture packets on, and choose a capture filter.
3. You can apply the capture filter by entering an expression next to the Capture Filter button. We want our filter to show only traffic inbound and outbound to port 262, so we enter **port 262**, as shown in Figure 4-10. (We'll discuss expressions in more detail in the next section.)
4. Once you have set your filter, click **Start** to begin the capture.

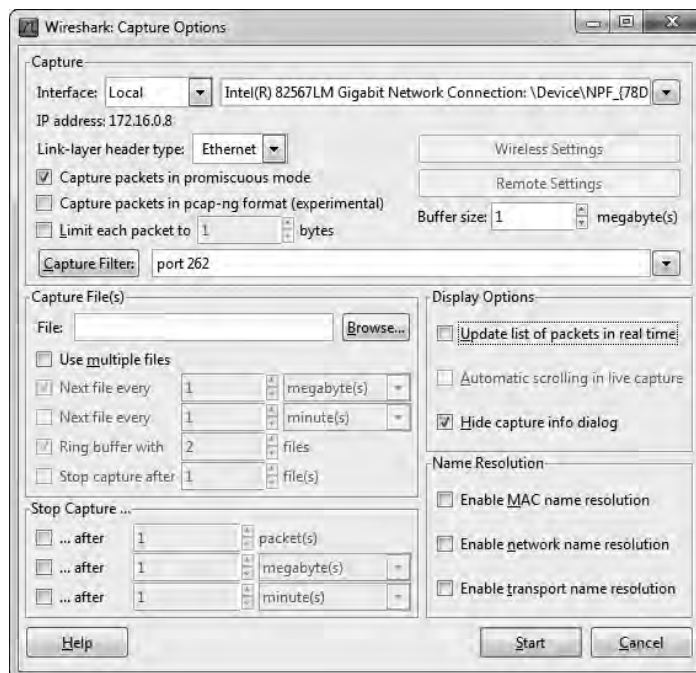


Figure 4-10: Creating a capture filter in the Capture Options dialog

After collecting an adequate sample, you should now see only the port 262 traffic and be able to more efficiently analyze this particular data.

Capture/BPF Syntax

Capture filters are applied by WinPcap and use the Berkeley Packet Filter (BPF) syntax. This syntax is common in several packet-sniffing applications, mostly because most packet-sniffing applications rely on the libpcap/WinPcap libraries, which allow for the use of BPFs. A knowledge of BPF syntax is 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 4-11.

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

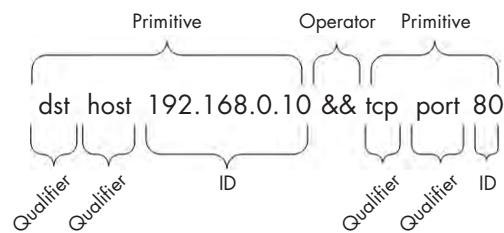


Figure 4-11: A sample capture filter

Given the components of an expression, a qualifier of `src` 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 source 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 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 its 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 like 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 leaving server 172.16.16.149 that is destined for a questionable host, 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, the equivalent of the preceding example could exclude that qualifier:

```
dst 172.16.16.149
```

Port and Protocol 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 based on services and applications that use known service ports. For example, here's a simple filter to capture traffic only on port 8080:

```
port 8080
```

To capture all traffic except that on port 8080, this will 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 powers 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, as follows:

```
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. Luckily, 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 compare that 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 6. 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. Multiple flags can be set simultaneously in a TCP packet, so we can't efficiently filter by a single `tcp[13]` value because several may represent the RST bit being set. Therefore, we must specify the location within the byte that we wish to examine by appending that location to the current primitive with a single ampersand (&). 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 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, 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 of the capture filters that I use most 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 (replace with your MAC)	Traffic to or from your MAC address
!ether host 00:00:00:00:00:00 (replace with your MAC)	Traffic not to or from your MAC address
broadcast	Broadcast traffic only
icmp	ICMP traffic
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 packet data without actually omitting the rest of the data in the capture file. That way, if you need to revert back to the original capture, you can simply clear the filter expression.

You might use a display filter to clear irrelevant broadcast traffic from a capture file; for instance, to clear ARP broadcasts from the Packet List pane when these 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, simply 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 Packet List pane, as shown in Figure 4-12. To remove the filter, click the **Clear** button.



Figure 4-12: Creating a display filter using the Filter text box above the Packet List pane

The Filter Expression Dialog (the Easy Way)

The Filter Expression dialog, shown in Figure 4-13, makes it easy for novice Wireshark users to create capture and display filters. To access this dialog, click the **Capture Filter** button in the Capture Options dialog, and then click the **Expression** button.

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

1. To view the specific criteria fields associated with a protocol, expand that protocol by clicking the plus (+) symbol next to it. Once you find the criterion you want to base your filter on, click to select it.
2. Choose the way that 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. When you've finished, click **OK** to view the completed text-only version of your filter.

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

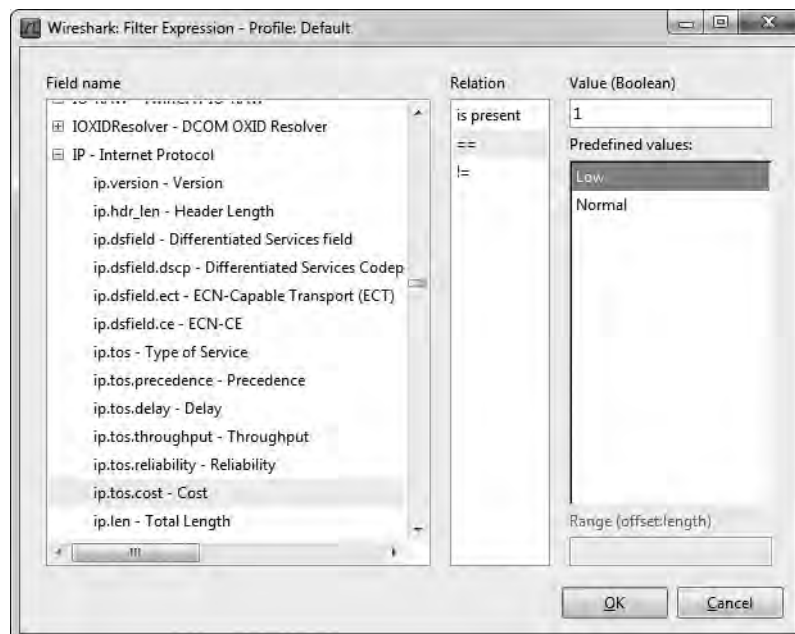


Figure 4-13: The Filter Expression dialog allows for easy creation of filters in Wireshark.

The Filter Expression Syntax Structure (the Hard Way)

You will most often use a capture or display filter to filter based on a specific protocol. 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 in length. You can use the "less than or equal to" operator (`<=`) to accomplish this goal in a filter expression like this:

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

Table 4-4 shows Wireshark's comparison operators.

Table 4-4: Wireshark Filter Expression Comparison Operators

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

Logical operators allow you to combine multiple filter expressions into one statement, dramatically increasing the effectiveness of our filters. For example, say that we're interested in displaying only packets to two IP addresses. We 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	Clear RDP traffic
tcp.flags.syn==1	TCP packets with the SYN flag set
tcp.flags.rst==1	TCP packets with the RST flag set
!arp	Clear ARP traffic
http	All HTTP traffic
tcp.port==23 tcp.port 21	Cleartext admin traffic (Telnet or FTP)
smtp pop imap	Cleartext 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 **New** button on the 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 **Save** button to save your filter expression in the list.

To save a custom display filter, follow these steps:

1. Select **Analyze** ▶ **Display Filters**, or click the **Filter** button above the Packet List pane to open the Display Filter dialog, shown in Figure 4-14.

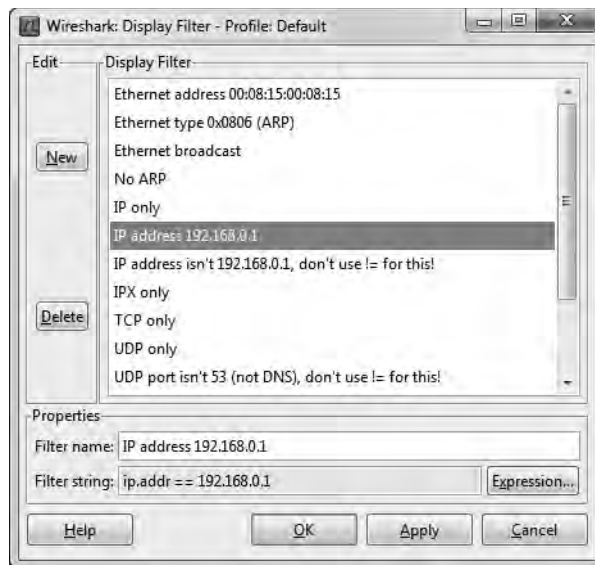


Figure 4-14: The Display Filter dialog allows you to save filter expressions.

2. Create a new filter by clicking the **New** button on the 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 **Save** button to save your filter expression in the list.

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