INSTANT

Short | Fast | Focused

Wireshark Starter

A quick and easy guide to getting started with network analysis using Wireshark

Abhinav Singh

[PACKT] PUBLISHING

Instant Wireshark Starter

A quick and easy guide to getting started with network analysis using Wireshark

Abhinav Singh

[PACKT] PUBLISHING

BIRMINGHAM - MUMBAI

Instant Wireshark Starter

First published: January 2013

Production Reference: 1180113

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84969-564-0

www.packtpub.com

Credits

Author

Abhinav Singh

Project Coordinator

Amigya Khurana

Reviewer

Sriram Rajan

Proofreader

Maria Gould

Acquisition Editor

Erol Staveley

Production Coordinator

Prachali Bhiwandkar

Commissioning Editor

Yogesh Dalvi

Cover Work

Prachali Bhiwandkar

Technical Editor

Veronica Fernandes

Cover Image

Sheetal Aute

About the author

Abhinav Singh is a young Information Security Specialist from India. He has a keen interest in the field of hacking and network security and has adopted this field as his full time employment. He is the author of *Metasploit Penetration Testing Cookbook*, *Packt Publishing*, which deals with Metasploit and penetration testing. He is also a contributor to the SecurityXploded community. Abhinav's work has been quoted in several portals and technology magazines. He can be reached at `abhinavbom@gmail.com`.

About the reviewer

Sriram Rajan is a Linux, FOSS, and Mac OS enthusiast. He has been using Linux since 2002. He started his career as a Systems Administrator (Solaris, Windows XP) in 2003. He has been working as Systems Software Engineer (C, Python, Linux) in the telecommunications industry. Currently he is employed as a consultant (C++, Linux) in the finance domain.

www.packtpub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

Table of Contents

Instant Wireshark Starter

Welcome to *Instant Wireshark Starter*. This book has been especially created to provide you with all the information you need to set up Wireshark and network analysis. You will learn the basics of Wireshark, get started with building your first course, and discover some tips and tricks for using Wireshark.

This book contains the following sections:

So, what is Wireshark? tells you what Wireshark actually is, what you can do with it, and why it's so great.

Installation teaches you how to download and install Wireshark with minimum fuss and then set it up so that you can use it as soon as possible on your favorite operating system.

Quick start – your first packet capture shows you how to perform one of the core tasks of Wireshark; network packet analysis. We will cover both the graphical as well as the command-line interface of Wireshark in this section.

Top 5 features you need to know about explains how to perform different tasks with the most important features of Wireshark. By the end of this section you will be able to:

- Start working with packet streams
- Understand name resolution and packet reassembling
- Analyze statistics of captured packets
- Decode captured data
- Export captured data
- Use Wireshark command-line tools

Wireshark activity shows live implementation of Wireshark and implements the topics mentioned previously.

People and places you should get to know provides you with many useful links to the project pages and forums, as well as a number of helpful articles, tutorials, blogs, and the Twitter feeds of Wireshark super-contributors.

So, what is Wireshark?

Wireshark is an open source network packet analyzer tool that captures data packets flowing over the wire (network) and presents them in an understandable form. Wireshark can be considered as a Swiss army knife as it can be used under different circumstances such as network troubleshoot, security operations, and learning protocol internals. This one tool does it all with ease.

Some of the important benefits of working with Wireshark are:

- **Multiple protocol support**: Wireshark supports a wide range of protocols ranging from TCP, UDP, and HTTP to advanced protocols such as AppleTalk.
- **User friendly interface**: Wireshark has an interactive graphical interface that helps in analyzing the packet capture. It also has several advance options such as filtering the packets, exporting packets, and name resolution.
- **Live traffic analysis**: Wireshark can capture live data flowing on the wire and quickly generate information about its protocols, flow media, communication channels, and so on.
- **Open source project**: Wireshark is an open source project and most of its development has been carried out through contribution from over 500 developers around the globe. We can write our own code and add to the project to meet our specific requirements.

These multiple functionalities of Wireshark make it one of the most popular open source network analyzer tools. In the later sections, we will discuss these operations of Wireshark in detail.

How does Wireshark work?

Let us give a brief introduction to the working process of Wireshark.

Network traffic sniffing is possible if the interface (network device) is transferred to promiscuous mode. This mode causes the interface to transfer all of the traffic it receives to the central processing unit rather than passing only the frames that the controller is intended to receive. Promiscuous mode was initially developed for bridged networking in virtualization.

Wireshark also works the same way. The entire process of network sniffing through Wireshark can be divided into three steps:

1. **Collection**: Wireshark transfers the network interface into promiscuous mode where it can capture raw binary data flowing on the wire.
2. **Conversion**: The chunks of binary data collected are then converted into a readable form. The packets are also re-assembled based on their sequence.
3. **Analysis**: The final step involves the analysis of captured and re-assembled data. The initial analysis involves identifying the protocol type, the communication channel, port numbers, and so on. At an advanced level, the different protocol headers can also be analyzed for a deeper understanding.

This was a quick introduction to Wireshark and its working methodology. In the next section we will cover its installation process in detail.

Installation

Let us start our journey to network analysis using Wireshark. First and foremost is to set up the Wireshark environment on our system. We will be covering both Windows-and Linux-based installation methodology and later discuss how we can set up a subversion environment to update different Wireshark libraries and dependencies. So let us start with setting up Wireshark on the Windows operating system.

In three easy steps, you can install Wireshark and set it up on your Windows system.

Step 1 – what do I need?

Before you install Wireshark, you will need to check that you have all of the required elements, listed as follows:

- Disk space: 100 MB free (min). You will require more free space to store captured packets.
- Memory: 256 MB (min), 1 GB (recommended).
- Wireshark requires a **network interface card** (**NIC**) that supports promiscuous mode.
- WinPcap driver that helps in packet capturing and sniffing.

Step 2 – downloading Wireshark

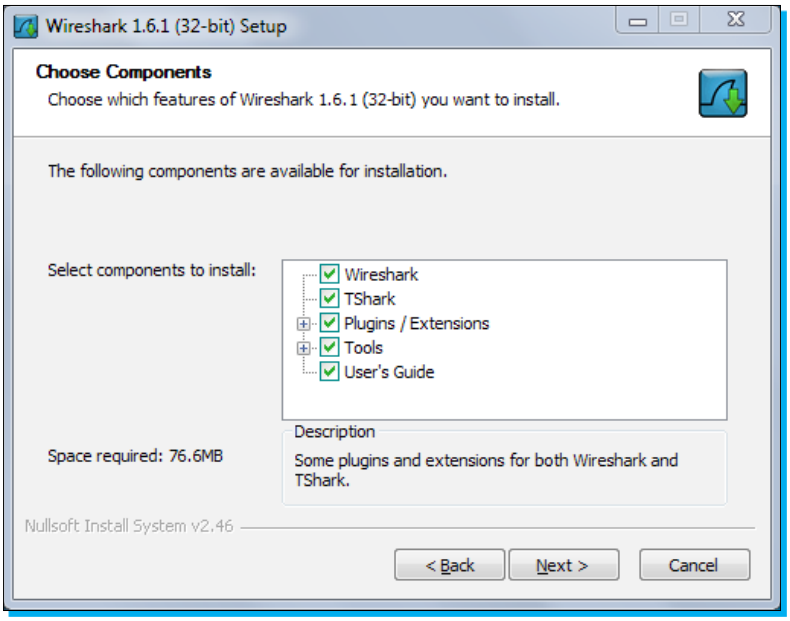
The easiest way to download Wireshark for Windows is to get a compressed package from `http://www.wireshark.org/`.

We suggest that you download the most current stable build according to your Windows version and architecture (x86 or x64). Windows users can identify their OS architecture by right-clicking on `My Computer`. Linux users can execute the `uname -i` command.

The following screenshot shows the Wireshark home page:

Step 3 - installing Wireshark

Once you have your choice of installer, you can follow the on-screen instructions to set up Wireshark on your system. It is a standard installer that will ask you to locate an installation directory, WinPcap installation, additional tools, and so on.



Wireshark comes bundled with the latest copy of WinPcap, so you don't need to manually set WinPcap. However, for your information, WinPcap can be downloaded from `http://winpcap.org`.

And that's it!

By this point, you should have a working installation of Wireshark and are free to play around and discover more about it.

Let us now move ahead and discuss setting up Wireshark on a Linux environment. The reason we are discussing Wireshark installation on Linux separately is that not all flavors of Linux are supported by the Wireshark project. You can find a complete list of supported Linux flavors on Wireshark's download page at `http://www.wireshark.org/download.html`.

Building Wireshark from source

To build Wireshark from its source files under Unix, you can follow these four steps:

Step 1 – getting the source files

Download the source package from the Wireshark download page (`http://www.wireshark.org/download.html`).

Step 2 – unpacking

Unpack the source from its gzip'd tar file using the following command:

```
gzip -dc wireshark-1.9-tar.gz | tar xvf
```

Step 3 – building

Change your current working directory to `wireshark`.

Step 4 – installing

Now we will have to build the source files into binary using the `make` command. Then the binary is installed onto the system using the `install` command.

```
root:~/wireshark-1#make
root:~/wireshark-1#make install
```

And that's it!

Your Wireshark is now ready to run on your Linux environment.

Installing Wireshark on Unix through binaries

Installing Wireshark through the binary is a simple process. You have to figure out your Unix type to get the correct binaries.

Installing from RPM

We can use the following command to install the Wireshark RPM binary downloaded from its website:

```
rpm -ivh wireshark-1.9.i386.rpm
```

Installing from DEB

To install Wireshark from the DEB binary, pass the following command to the terminal window:

```
apt-get install wireshark
```

```
root@bt:~/Desktop# apt-get install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
wireshark is already the newest version.
The following packages were automatically installed and are no longer required:
  libdmraid1.0.0.rc16 python-pyicu libdebian-installer4 cryptsetup libecryptfs0 reiserfsprogs rdate
  libdebconfclient0 dmraid keyutils
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@bt:~/Desktop#
```

Many Linux versions ship installed copies of Wireshark. You can look for a package update using `apt-get update` to look for new versions.

Setting up the subversion client

Setting up the subversion client is an optional topic for those who want to set up the source environment of Wireshark. Subversion can help in the quick update of code files and libraries. You can set up any subversion software of your choice. Here we will take the example of Tortoise SVN which is a popular open source subversion client. You can download the setup from `http://tortoisesvn.tigris.org/`. Once you are through with the setup, right-clicking on any folder will show the SVN options.

To set up the subversion for Wireshark, follow these simple steps:

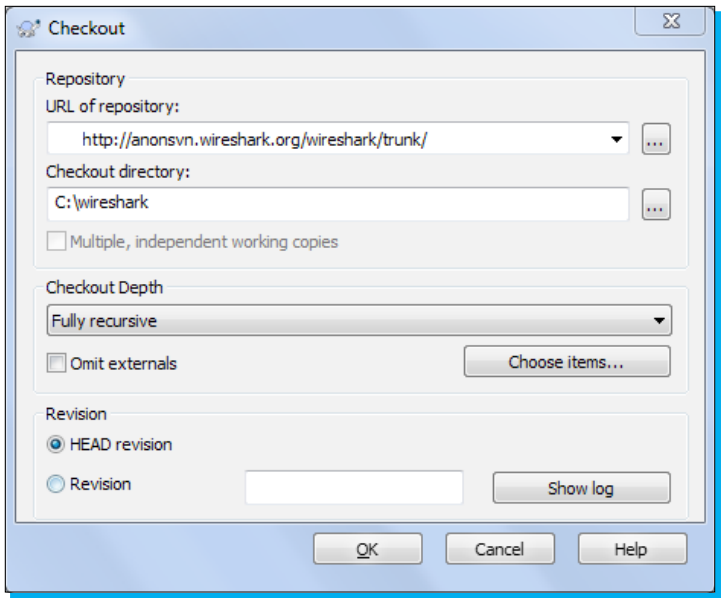
Step 1 – creating the directory

Create a new directory/folder with the name **wireshark**. Right-click on the folder and move to **svn checkout**.

Step 2 – setting the subversion path

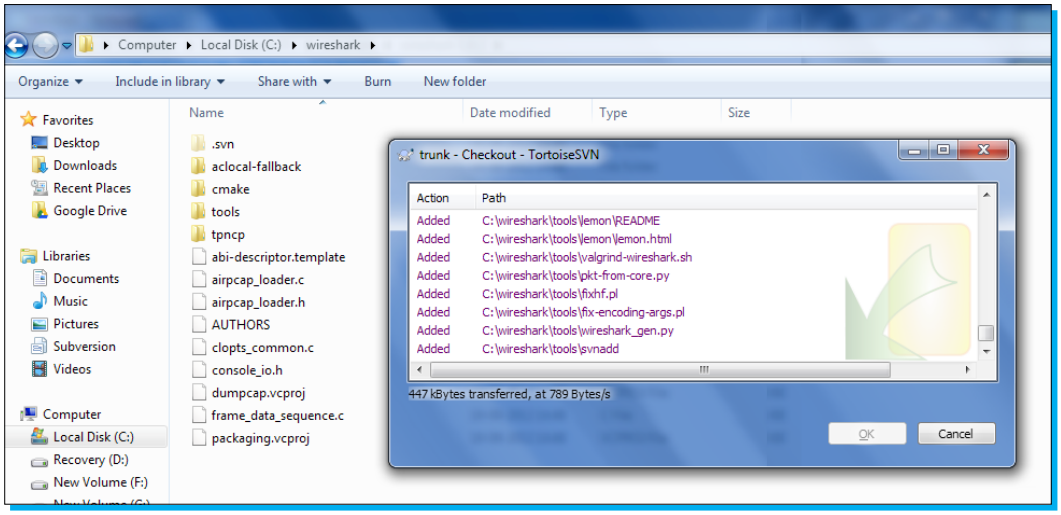
Under **Url of Repository** enter **http://anonsvn.wireshark.org/wireshark/trunk/**.

Under **Checkout directory**, make sure that it reflects the same path where you have created your Wireshark directory. Click on **OK** to start the update process.



Step 3 – checkout

Once the subversion starts populating your `wireshark` folder, you will see different source directories getting created.



Now that your tortoise client has been set up, you can right-click on the `wireshark` folder and select **SVN update** to get updated copies of the source code any time. This reduces the overhead of manually downloading the new updates.

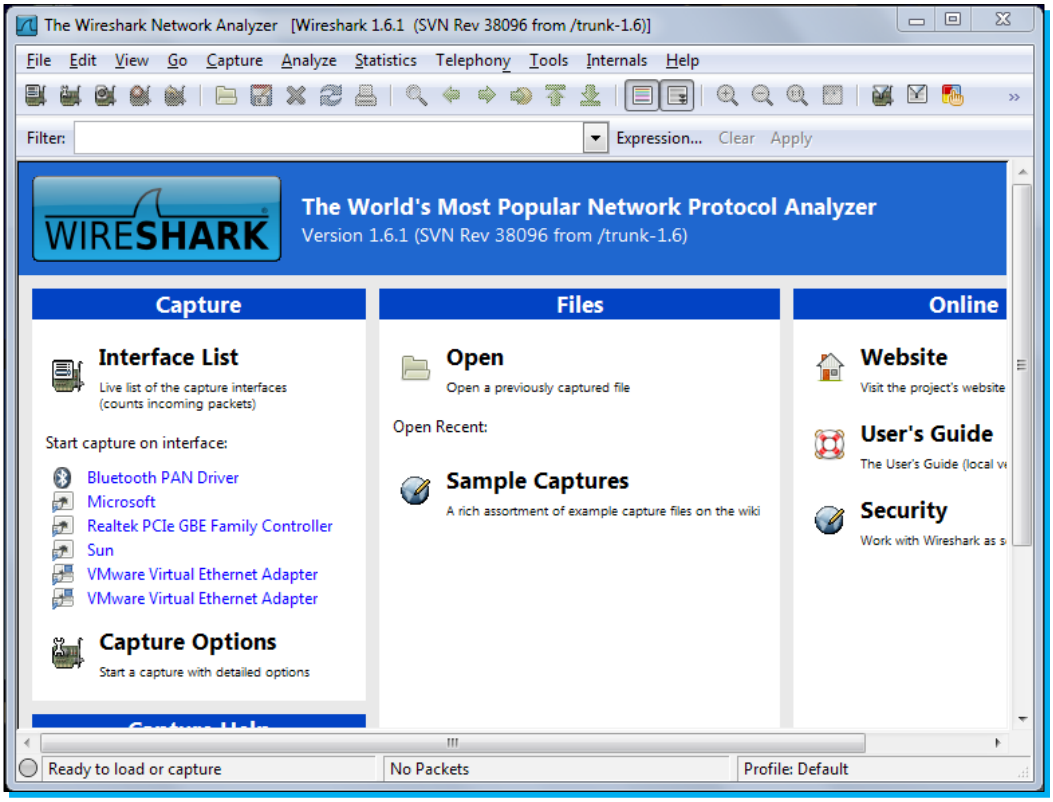
This was a quick guide to setting up Wireshark under different environments. In the next section we will see how to start working with Wireshark and analyze our first packet capture in detail.

Quick start – your first packet capture

Now that we have set up Wireshark on our system, we can move ahead and start experimenting with its features. In this section we will cover some of the basic features and quick tips that are essential for getting started with packet capture using Wireshark. We will start with the basics of Wireshark where we will take a brief look at its GUI and later on we will experiment with packet capture and the analysis of the captured data. Meanwhile we will be using some common network protocols and terminologies such as HTTP, TCP, and data packets. Familiarity with these terms can help in a better understanding of packet capturing. So let us move ahead to start our journey with Wireshark.

Getting started with network interface selection

The first and foremost thing to start with is selecting a network interface on which you want to capture the data. Once we have set up Wireshark on our system, we can launch it from the desktop or start menu or through the command line depending on your operating system. The first thing that Wireshark will prompt is to select a network interface. A typical Wireshark launch panel will look similar to the following screenshot:

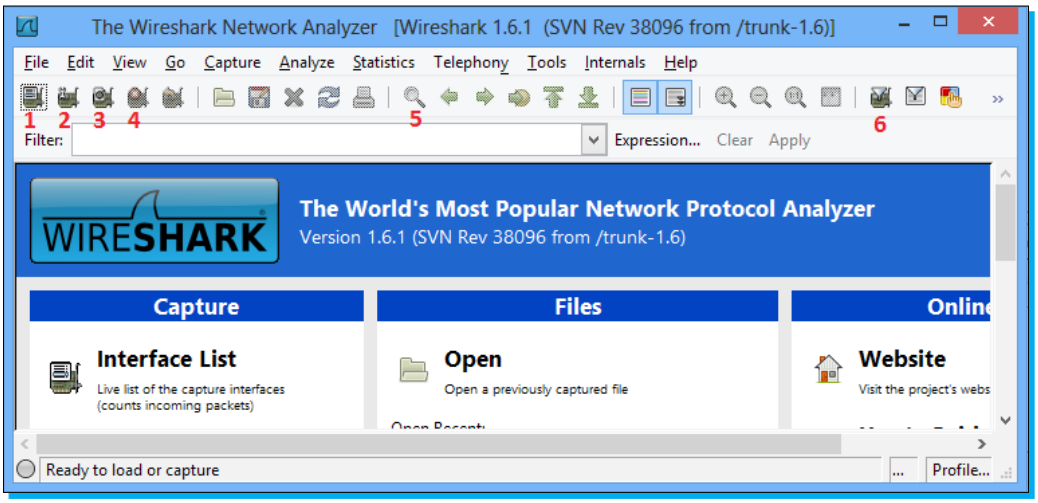


As you can see, the top-left column of the main window displays different capture interfaces under the heading **Interface List**. We can select any interface of our choice to start working with. For example, to capture the LAN traffic flowing across your system, you can choose the default LAN network card installed on your system. Similarly you can select the 802.11 Ethernet adapter for wireless data capture over LAN and so on.

Once we are through with the network interface selection, we can move ahead with packet capturing but before jumping to it, let us take a quick look at the Wireshark GUI and understand the functionality of some of the useful menu items.

A quick look at the Wireshark GUI

Looking at the previous screenshot, you can see that the main menu bar of Wireshark contains some of the commonly known menu items such as **File**, **View**, **Edit**, and **Help**. The other menu items such as **Analyze** and **Capture** will be discussed later in other sections of the book. Below the main menu bar, we have specific menu icons which are used for the quick launch of common actions performed during packet capture and analysis. Let us take a brief look at some of them.



- **List available capture interfaces (1)**: This menu icon is used to change or select a new interface media while working with packet capture.
- **Show capture options (2)**: This icon launches a mini panel to customize the data capture settings. Some of the main customizations that can be made are:
  - Changing the capture type
  - Setting up the buffer size for capture
  - Limiting the size of captured data
  - Managing display options and name resolution

- **Start a new live capture (3)**: This icon is used to launch a fresh capture from the selected interface.
- **Stop the running of live capture (4)**: This icon is used to stop the current live capture while maintaining the captured data in the buffer for further processing.
- **Find a packet (5)**: This icon is used to look for a particular text/string/parameter within the captured packets.
- **Edit capture filter (6)**: This icon is used to modify the capture filter applied to data packets. We will cover this in detail in our next section.

You will also notice a **Filter** box under the menu icons. This box is used to quickly apply a particular filter over the captured packets. For example, we can view only the DNS request/response by typing `dns` in the **Filter** box. It also reflects the current display filter that is applied on the captured traffic.

Wireshark GUI panels

Let us now take a quick look at the different panels present in the Wireshark GUI. Typically we can divide the GUI panels into four parts: capture panel, packet details panel, packet bytes panel, and lastly the status panel. We will go through each of these one by one.

Capture panel

The capture panel displays the live capturing of network packets in a sequential order. Each line in this list reflects a single captured packet. This intelligent display panel divides the information into rows and columns. Each row represents a single data packet whereas each column represents additional information about the packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CadmusCo_1a:8b:91	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10
2	0.000478	RealtekU_12:35:02	CadmusCo_1a:8b:91	ARP	60	10.0.2.2 is at 52:54:00:12
3	0.000494	10.0.2.15	202.56.240.5	DNS	80	Standard query A download.
4	0.084935	202.56.240.5	10.0.2.15	DNS	253	Standard query response CN
5	0.086205	10.0.2.15	63.245.217.39	TCP	62	neod1 > http [SYN] Seq=0 W
6	0.343490	10.0.2.15	63.245.217.39	TCP	62	neod2 > http [SYN] Seq=0 W
7	0.404601	63.245.217.39	10.0.2.15	TCP	60	http > neod1 [SYN, ACK] Se
8	0.404645	10.0.2.15	63.245.217.39	TCP	54	neod1 > http [ACK] Seq=1 A
9	0.432101	10.0.2.15	63.245.217.39	HTTP	516	GET /?product=firefox-15.0
10	0.433180	63.245.217.39	10.0.2.15	TCP	60	http > neod1 [ACK] Seq=1 A
11	0.674462	63.245.217.39	10.0.2.15	TCP	60	http > neod2 [SYN, ACK] Se
12	0.674511	10.0.2.15	63.245.217.39	TCP	54	neod2 > http [ACK] Seq=1 A
13	0.825277	63.245.217.39	10.0.2.15	HTTP	544	HTTP/1.1 302 Found
14	0.942671	10.0.2.15	63.245.217.39	TCP	54	neod1 > http [ACK] Seq=463
15	1.364269	10.0.2.15	202.56.240.5	DNS	84	Standard query A download.

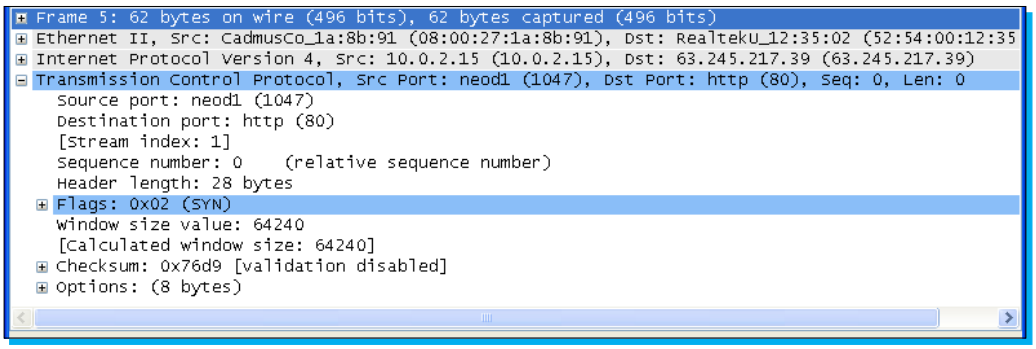
The columns are as follows:

- **No.**: This represents the packet sequence number to identify packets uniquely
- **Time**: This represents the time stamp when a packet is captured
- **Source**: This represents the IP address/device from where the packet is coming
- **Destination**: This represents the IP address/device where the packet is going to
- **Protocol:** This represents the protocol type of the captured packet
- **Length**: This represents the size of the packet
- **Info**: This represents quick additional information about the packet

Each protocol is represented using unique coloring schemes in Wireshark. This enables the user to easily distinguish between different protocol types.

Packet details panel

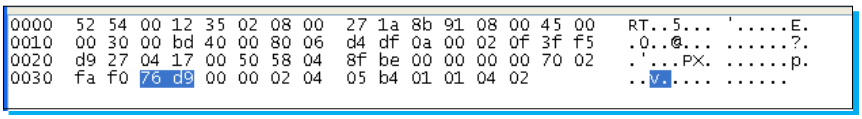
Whenever a single data packet is selected from the capture panel, its detailed information is shown inside the packet details panel.



It contains detailed information about the protocols and its different parameters in a tree structure which can be expanded and collapsed. This information can be helpful in network forensics.

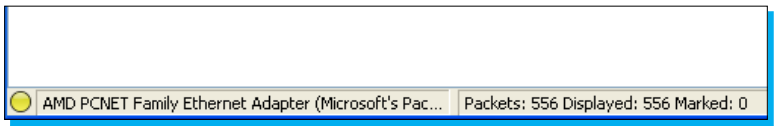
Packet bytes panel

The packet bytes panel represents the information of the packet details panel in a dump or actual format. It shows the byte sequences of the flow.



Here also the information is divided into three columns where the first column represents the data offset, the next column represents the data in hexadecimal values, and the last column represents the ASCII representation of information.

The status panel shows the current status of our operation. It reflects information such as the capture status, count of packets (captured, displayed, and/or marked), and the file location where the captured packets are stored.



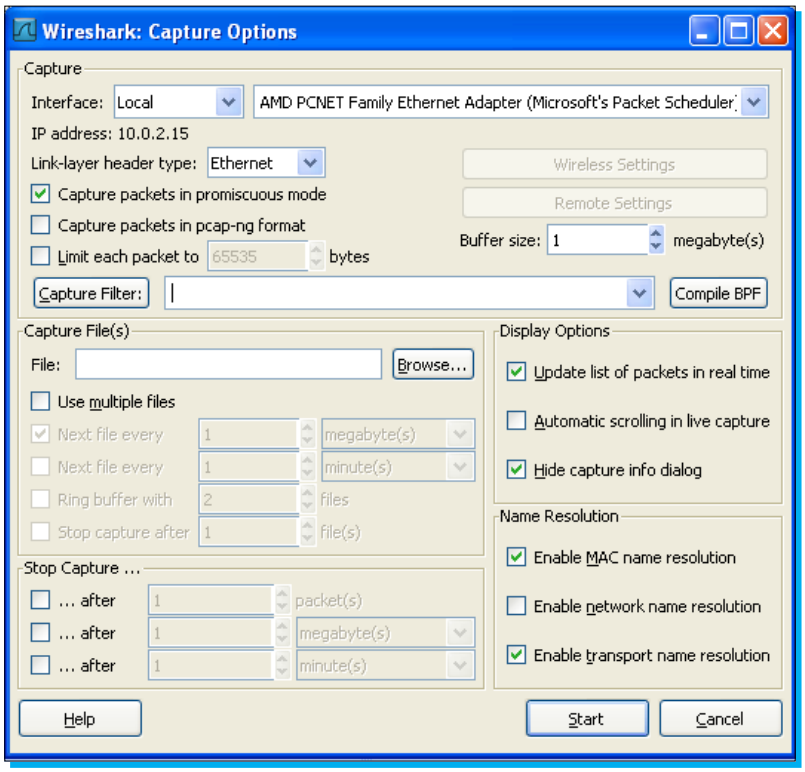
We took a quick look at some of the important features in the Wireshark GUI. We will now proceed with some technical aspects of our packet capturing tool.

Setting up filters

Here we will cover one of the most technical and useful discussions of packet capture. Filters play a very important role in packet capture. While working on a LAN or while capturing the packets on a server hosting many services, we can face problems in monitoring a particular protocol or service. To remove this overhead we use filters. Filters can be applied at two ends, namely capture filters and display filters. Let us start with capture filters.

Capture filters are applied to monitor packets selectively. It will filter out or capture only that traffic which is assigned by us. To do this selective capturing we will have to pass the `winpcap` command instructions to Wireshark.

The **Capture** filter option can be launched by clicking on the Edit Capture Filter menu icon. Alternatively, it can also be launched by clicking on **Capture | Options**. You will see an option window similar to the following screenshot:



We can enter our filter options by filling in the **Capture Filter** field. Alternatively, we can also click on **Capture Filter** and store our capture filter rule for future rules.

The question that now arises is how to write a filtering rule. WinPcap rules for packet capture follow a definite pattern. A typical structure for writing a rule can be as follows:

<Protocol name><Direction><Host(s)><Value><Logical operations><Expressions>

For example, to capture TCP packets when the source port is 443, we will write the following rule:

```
tcp src port 443
```

Similarly, to drop ARP packets we can use `not arp`.

To capture both inbound and outbound traffic on port 80 (http), we can use `port 80`.

To capture packets where the source IP is 192.168.56.101 and the port number is 232 we can use `src 192.168.56.101` and port `232`.

In this way we can combine different conditions to create our own capture filters and reduce the overhead. If no capture filter is applied, then all the network packets flowing through the selected interface are captured by Wireshark.

The next filter option is the display filter. It is used to select particular packets from the captured file. Unlike capture filters, display filters can be applied even after the packets have been captured. The display filter menu can be launched by clicking on the Edit/Apply Display filter menu icon. Alternatively, display filter rules can also be applied using the filter bar available on the main GUI window.

Display filter rules also follow a fixed structure:

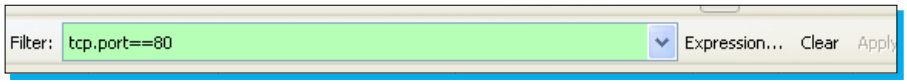
<Protocol> . <String 1> . <String 2> . <Comparison Operator><Value><Logical operators><Expressions>

Let us pick up some examples to implement this rule structure practically.

To view the TCP packets captured on port 80, we can use the following display filter:

```
tcp.port==80
```

Typing `!arp` and pressing *Enter* in the **Filter** bar will drop all the ARP packets.

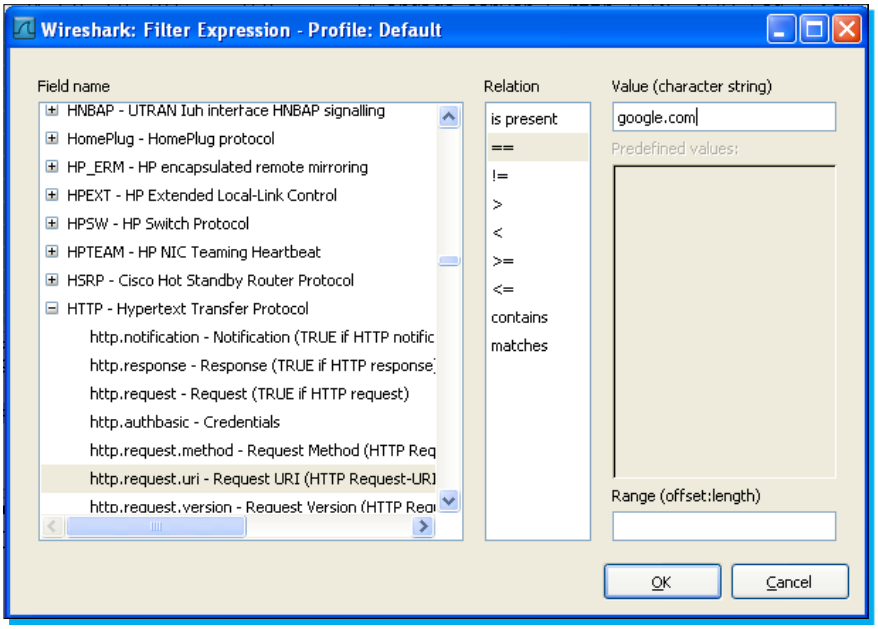


The filter `ip.addr==192.168.56.101` will display packets only from a particular IP.

Now that we have covered the hard part about packet filters, we will discuss a quick and easy way to perform both types of filter options.

Working with the Filter Expression dialog box

The **Filter Expression** dialog is a feature that makes it easy for novice Wireshark users to create, capture, and display filters. It can be launched by clicking on the **Expression...** button present on the **Filter** expression bar (refer to the previous screenshot).



Using this we can easily create display/capture filters as it provides us with multiple options along with some pre-loaded expressions. To create an expression you can follow these simple steps:

1. To view the specific criteria fields associated with a protocol, expand that protocol by clicking on the plus [⊞] icon next to it.
2. Once you find your desired criteria, you can select the relation such as equal to (==) or greater than (>).
3. Next you can provide the value with which you want to compare (for example, `google.com` or a number).

If you notice the previous screenshot, the fieldname **http.request.uri** is selected to check if it is equal to (Relation) **google.com** (Value). Hence the complete filter expression becomes `http.request.uri==google.com`.

This was a quick demonstration of using the **Filter Expression** box. In the next section we will start working with our first packet capture.

Capturing live data

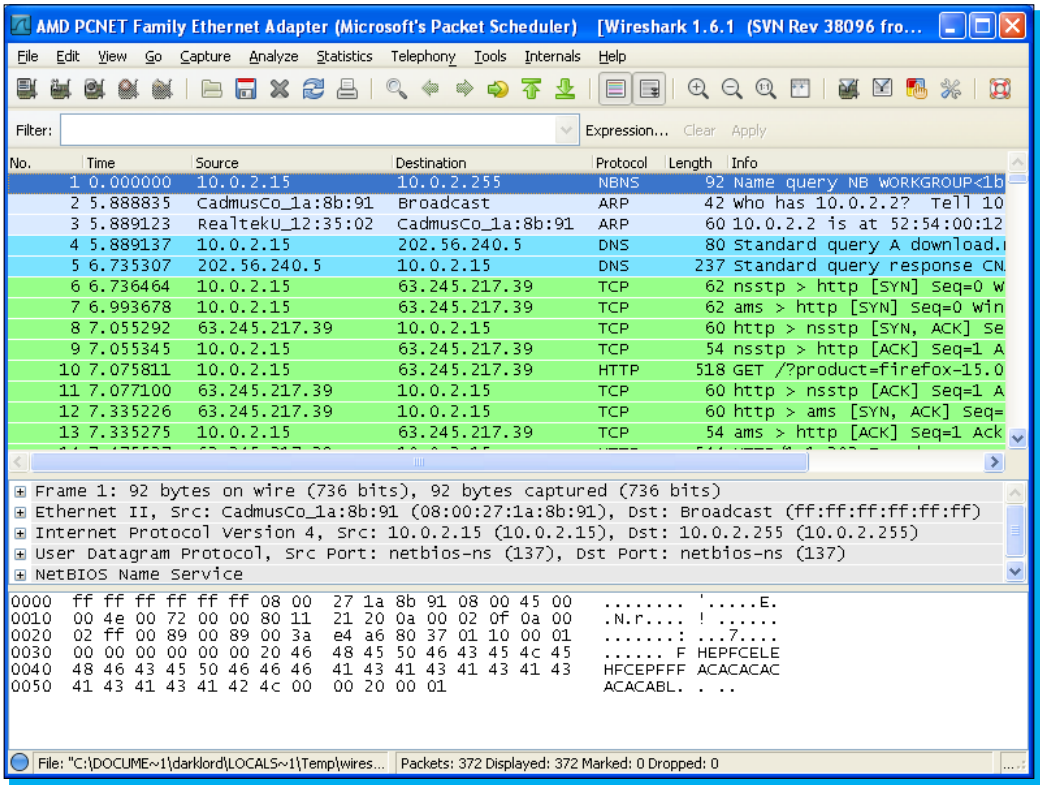
Now that we have developed enough background about Wireshark, we can start with the "Hello World" of packet capturing. In this section we will take a quick look at how we can start with capturing packets using Wireshark.

To start capturing data packets in a Windows environment, follow these simple steps:

1. Launch Wireshark from the start menu.
2. Apply the desired capture filters.
3. Choose a network interface to start capturing the traffic.

These same steps can be repeated for launching Wireshark in Linux-based operating systems as well. The only difference lies in selecting the network interfaces as Linux shows network devices instead of network description.

Once you have selected the interface, you will notice that the capture panel starts populating with captured packets.



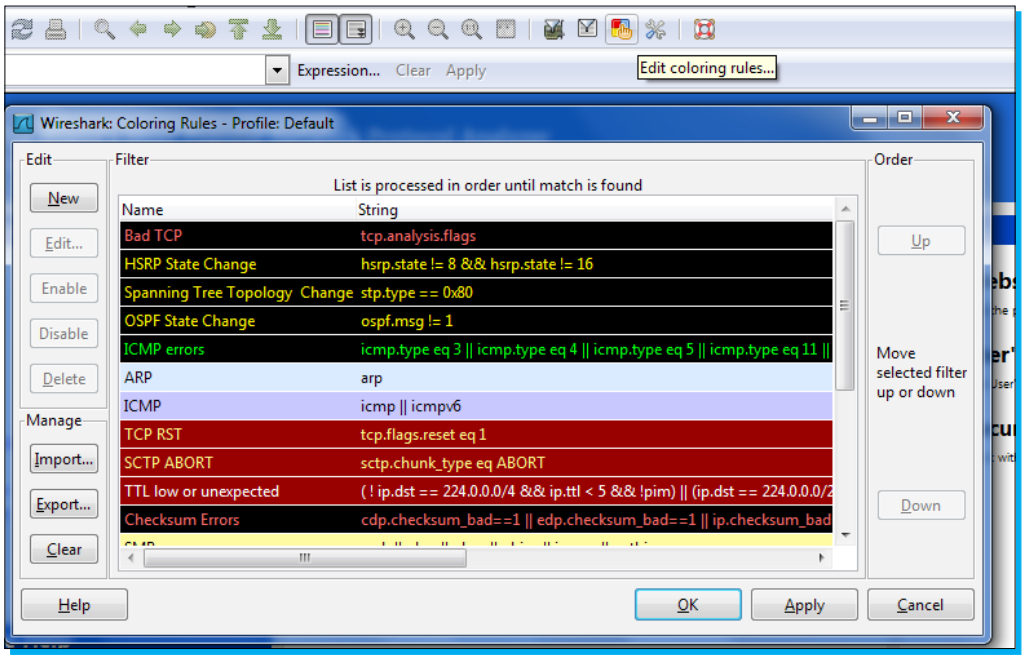
You can stop the live capture at any time by clicking on the **Stop** menu icon. A new live capture can be started by clicking on the **Restart live capture** option. Note that the currently captured data will be erased if we select the restart option.

You will notice different coloring schemes used by Wireshark to mark different protocols. Let us move ahead and take a quick look at this coloring scheme and how it is implemented.

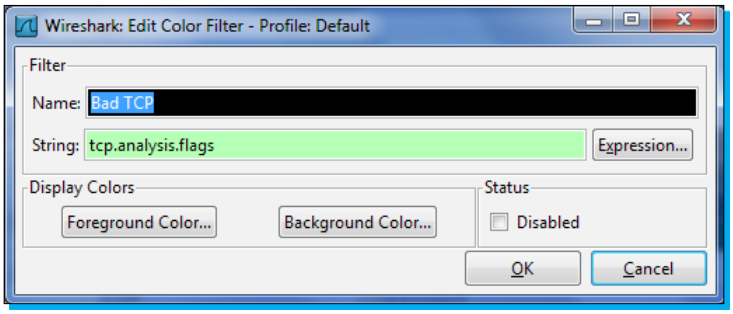
Understanding the Wireshark coloring scheme

You might have noticed by now the colorful scheme that Wireshark uses to distinguish different protocols. In fact the coloring scheme is used in case of a bad packet, checksum error, and other common packet errors that may occur in a network.

To take a closer look at the coloring scheme, click on the **Edit coloring rules...** from the menu icons. This will launch a menu box reflecting the default coloring scheme.



You will notice different default coloring patterns used to represent protocols, errors, failures, and so on. The default scheme can also be changed to being user specific by clicking on the **Edit** button.



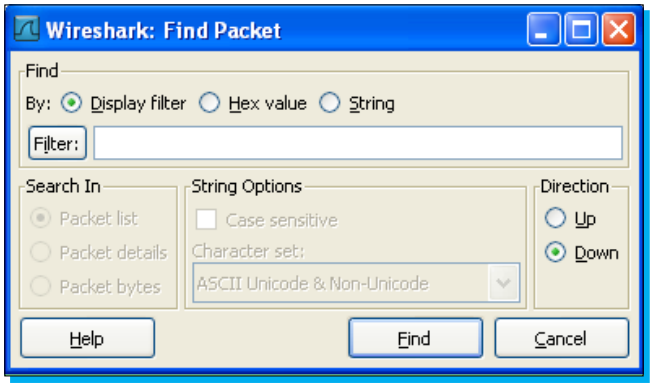
This will provide you with the flexibility to change the different coloring schemes such as background color, foreground color, name to represent the protocol, and string pattern.

Working with captured packets

The main reason for packet capture is to analyze the network activity. Captured packets can be analyzed very effectively using Wireshark. Let us continue from our previous discussion where we learnt about performing a live capture. Let us click on the **Stop live capture** menu icon and begin our experimentation with captured packets.

Searching for packets

Searching for specific information can be carried out by navigating to **Edit | Find Packets** or by pressing the *Ctrl + F* keys. This will launch a search box with three different search options, namely **Display filter**, **Hex value**, and **String**.



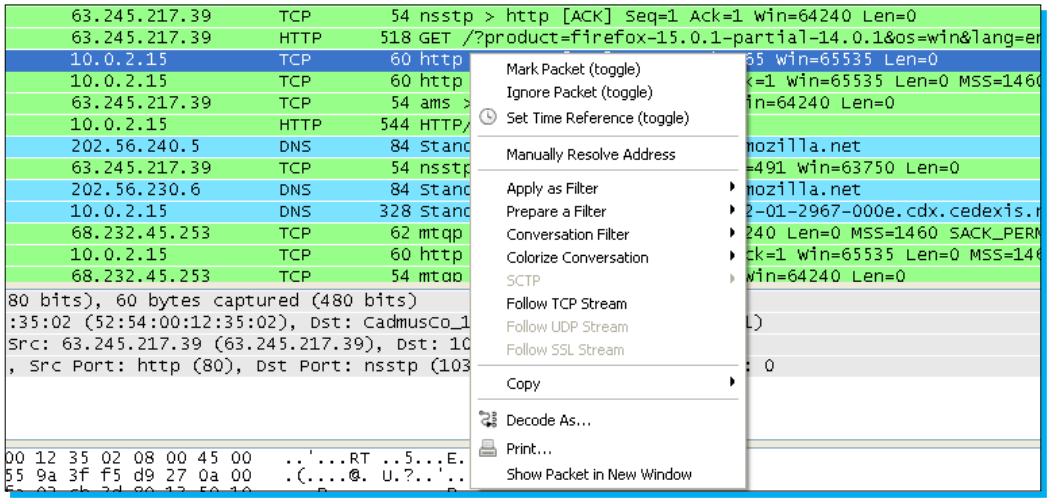
The **Display filter** option allows us to enter an expression-based filter that will find only those packets that satisfy that expression.

The **Hex value** search option is used to look for a specific hexadecimal sequence.

The **String** search option allows us to look for specific strings. Searching for specific strings activates another supporting search option listed under the **Search In** table. To search for a particular string from the list of captured packets, we can select the **Packet list** option. To look for a particular string type that can exist in the packet header information, we can use the **Packet bytes** option. Finally to search for a string inside the data contained by the packet, we can use the **Packet details** option.

Marking packets

Marking important packets can be useful in quickly retrieving the required information. This can be helpful in keeping track of important packets in a huge list of captured data. To mark any packet, right-click on it and select **Mark Packet (toggle)**. The background color of the marked packet changes to solid black for visual identification.



Saving captured data

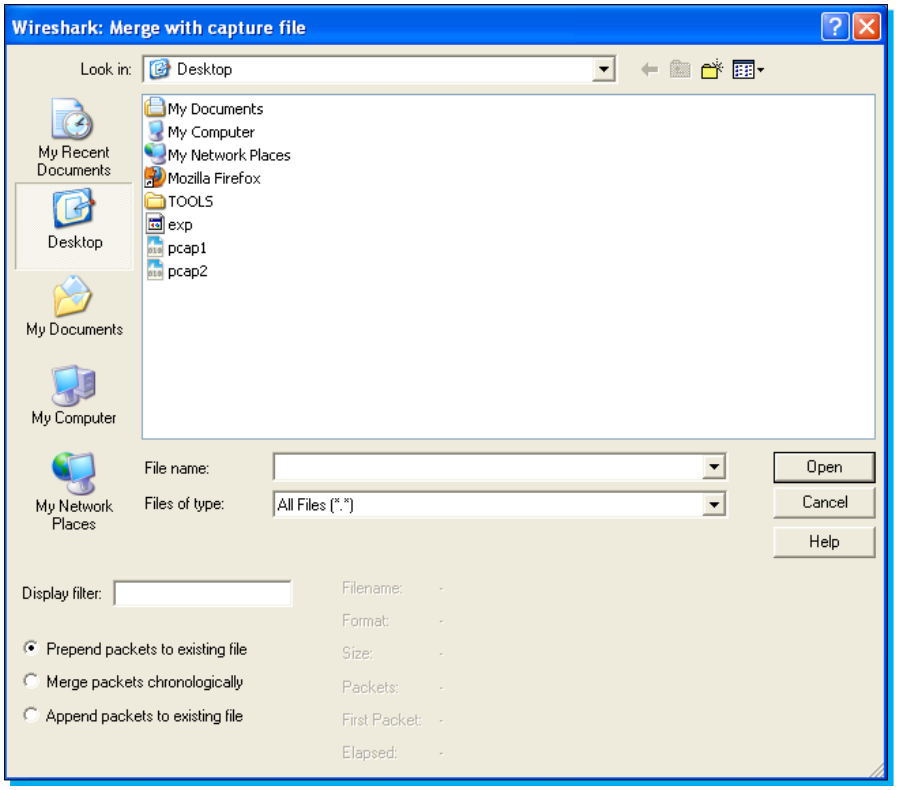
To save the captured file, go to **File | Save as**. Here we will have an option to either save all the captured packets or selectively save only the displayed packets after applying some display filter. Wireshark supports various packet capture file formats such as `.pcap`, `.cap`, and `.pcapng`.

Exporting and merging packets

Wireshark also allows the exporting of the captured packets into different formats such as `.txt`, `.csv`, and `.xml`. This feature is helpful when the data is to be used in other operations. Packets can be exported by going to **File | Export | File**.

Wireshark also provides the feature of merging different capture files into one. To merge a capture file, follow these steps:

1. Open one of the capture files you want to merge.
2. Choose **File | Merge** to bring up the **Merge with Capture File** dialog.



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.

Printing packets

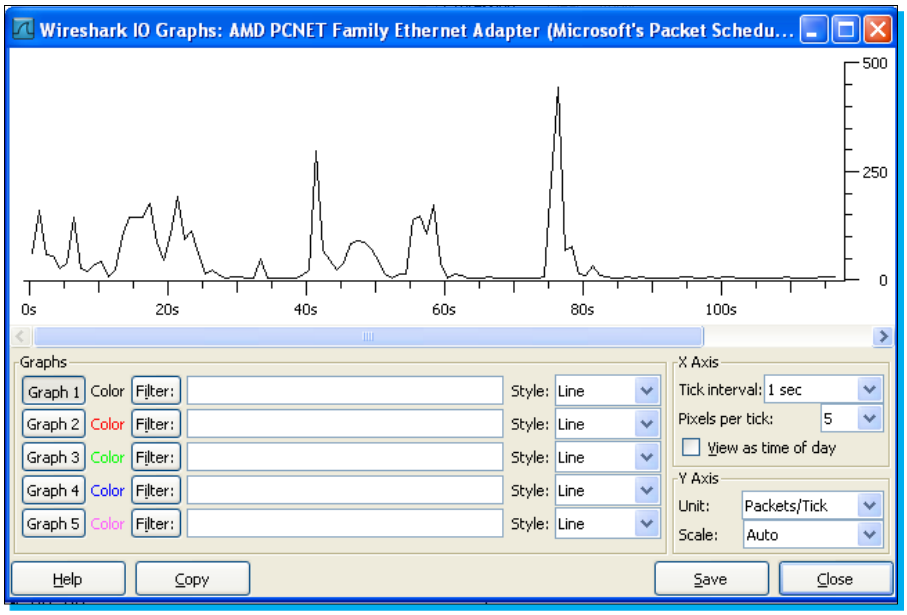
Wireshark allows the printing of captured packets as plain text, post script, or to an output file depending on your choice. It also allows you to provide a specific range of packet numbers for selective printing. Navigating to **File | Print** will launch the print box.

This was a brief demonstration of working with captured packets to enhance productivity. In the next section we will focus on some other lesser known features of Wireshark which can be essential during packet forensics and network penetration testing.

Double clicking on any packet will open a separate popup window displaying information about it.

Input/Output graph window

Wireshark also provides a cool feature to quickly develop a graphical overview of our captured packets. This can be helpful in monitoring the amount of data flowing across the network. Graphical analysis can also be helpful in analyzing large amounts of data. To generate a graphical view of your captured file, go to **Statistics | IO Graph**.



There are different operations that can be performed with this graph.

Graphs

We can enable five different comparison graphs and provide them with different coloring schemes for easy viewing.

Filter

We can also filter out packets from our analysis to build specific graphs. Clicking on **Filter** will open the same menu box which we have seen earlier in our *Setting up filters* section.

Style

Style provides us with the flexibility to choose from different graph options that we can build (Line/Impulse/FBar/Dot).

Graph co-ordinates

The graph co-ordinates, that is, the **X Axis** and the **Y Axis**, can also be modified accordingly.

X Axis

- **Tick interval**: An interval in the X direction lasts (10/1) minutes or (10/1/0.1/0.01/0.001) seconds
- **Pixels per tick**: Uses 10/5/2/1 pixels per tick interval
- **View as time of day**: Gives the option to view the X direction labels as time of day instead of seconds or minutes since beginning of capture

Y Axis

- **Unit**: The unit for the Y direction (Packets/Tick, Bytes/Tick, Bits/Tick, and Advanced)
- **Scale**: The scale for the Y unit (Logarithmic, Auto, 10, 20, 50, 100, and so on)

You can click on any graph region/interval to view the packet stream which is flowing during that time period.

Copying and saving

The **Copy** option can be used to copy the values of packets directly from the graph and paste them into a CSV file. A typical CSV file will look similar to the one shown as follows:

	A	B
1	Interval start,"Graph 1"	
2	0.000,"58"	
3	1.000,"158"	
4	2.000,"55"	
5	3.000,"54"	
6	4.000,"23"	
7	5.000,"38"	
8	6.000,"142"	
9	7.000,"24"	

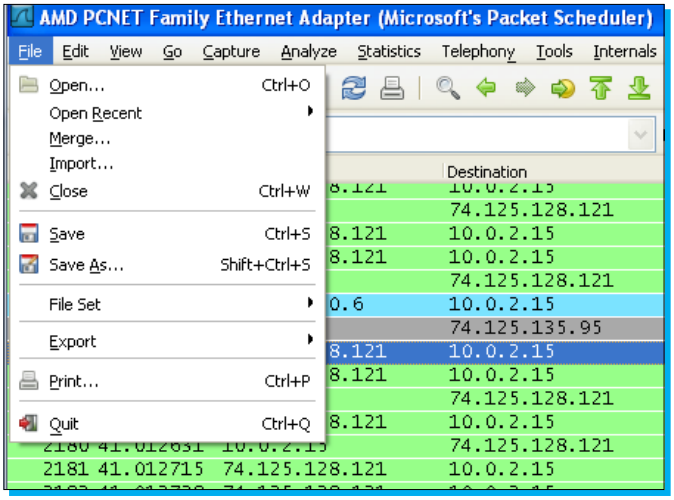
The **Save** option enables us to save the graph in several image formats such as JPG, PNG, and BMP.

File input/output

We have already seen several operations on data packets in the *Working with captured packets* section. Here we will throw some light on various file formats and different importing/exporting techniques that can be helpful for us in understanding Wireshark in detail. So let us start with a basic discussion of opening a captured data packet.

Opening captured packets

To open the captured packet, move to **File | Open**. This will launch the file selection panel from where we can select the file we want to open.



In this way we can open and perform various operations on the captured packets which we have discussed so far. We have already discussed about saving and merging different captured files in our previous sections. An important thing to discuss here is the different file formats supported by Wireshark. Let us take a quick look at them.

Wireshark file formats

Wireshark supports a wide range of file formats to open or save data packets. This is one of the major reasons which makes Wireshark the most popular packet capture tool. Packets saved from other tools can also be opened inside Wireshark, and it can save the captured packets in several formats so that other tools can also understand and analyze them. We will look at different input and output formats supported by Wireshark in this section.

Input file formats

Wireshark supports over 30 different tools to seamlessly open their captured files. Some of them are:

- Libpcap: This is the most widely used file format for network captures. Products which save captured network packets as libpcap are Wireshark, tcpdump, psniff, and so on
- Microsoft network monitor (Netmon): This is a network capture tool from Microsoft
- Network Associates Windows-based Sniffer: Files generated using Microsoft sniffers
- Symbian OS btsnoop captures: These are Symbian-based sniffing tools
- IBM Series (OS/400) Comm traces: These are sniffing tools designed to work with IBM devices
- Pcap-ng: This is the successor of libpcap with several modifications and added features

Some of the other vendors supported by Wireshark include Juniper, HP, Toshiba, and RADCOM.

Now let us move to some of the important file formats in which Wireshark can save the packets.

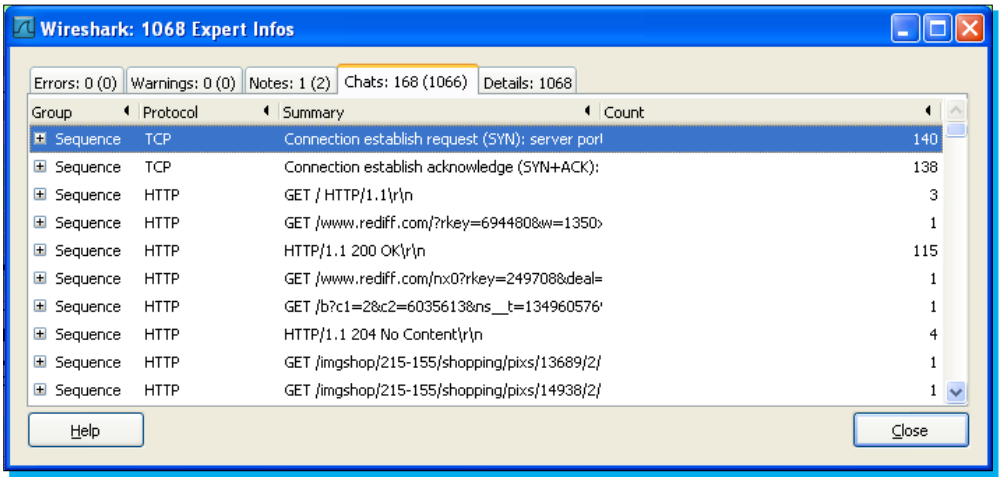
- libpcap, tcpdump, and various other tools using tcpdump's capture format
- Microsoft Network Monitor - NetMon (*.cap)
- Network Associates Sniffer (*.cap)
- Pcap-ng

This was a quick reference to some of the important file formats supported by Wireshark. This information can be helpful during file import and export. Let us move ahead with another important discussion about file input/output.

Expert Infos

Expert Info is another powerful feature of Wireshark that helps in keeping track of any abnormality that may arise in packet capture. It keeps a log of all the packets and provides an approximate intelligent analysis. The information includes reporting, error, failure, protocol, and severity. Let us take a close look at how we can leverage the power of Expert Infos.

To launch the Expert Infos dialog, go to **Analyze** | **Expert info Composite**.



You will notice five different tabs at the top. Let us define their purpose one by one:

- **Errors**: This tab reports about any error that may have a risen during packet capture
- **Warnings**: This Expert Info gives warning information such as segmentation error and out of order packets
- **Notes**: This tab gives approximate information in case of duplicate packets, protocol issues, and so on
- **Chats**: This tab gives us information about communication calls such as TCP GET request and application calls

We can view the packets any time by expanding the message and clicking on the packet number. The capture window will automatically point to that packet.

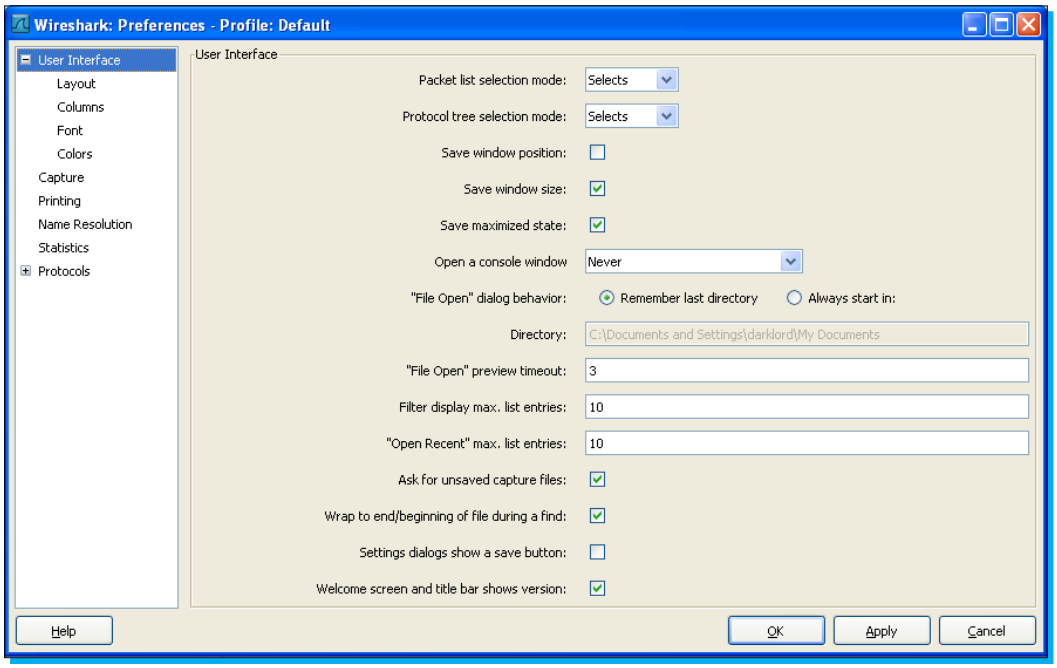
Each tab discussed earlier contains a separate information table that displays information contained in the tabs. This information table contains four columns, namely Group, Protocol, Summary, and Count. Some older versions of Wireshark can also contain a fifth row named Severity. Let us define them in a nutshell.

- **Group**: This Expert Info tells us about the category under which the information has been categorized. The different categories include checksum, sequence, and request/response code.
- **Protocol**: This represents the protocol for which the Expert Info has been reported.
- **Summary**: This gives short additional information about the Expert Info.
- **Count**: This represents the packet position in the entire file.

This is a quick demonstration of using Expert Infos for better network analysis. But this is not the only way in which Wireshark provides expert analysis. We can also view this information directly on the main capture panel. To add to this column, we will have to use the **Preferences** option. Let us discuss this in detail in our next section.

Using preferences

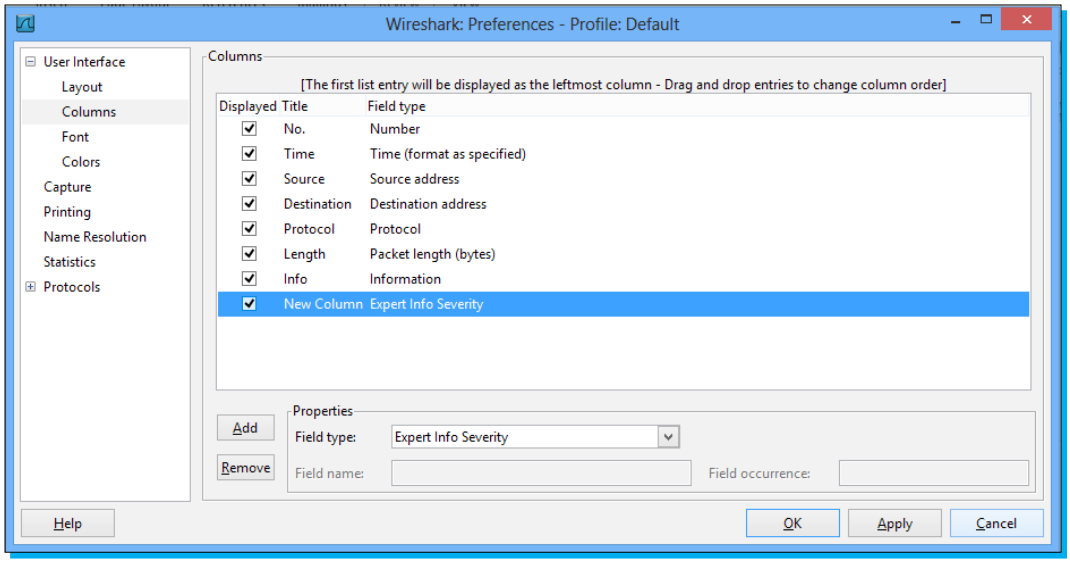
Preferences can be used to set up our own display and monitor options. This enhances the capability of Wireshark beyond the default settings and the user can directly view those things which are important for his/her analysis. To bring up the **Preferences** box, navigate to **Edit | Preferences...**.



You can modify and add several different view options from this window. Let us move ahead and add the expert column to our capture panel.

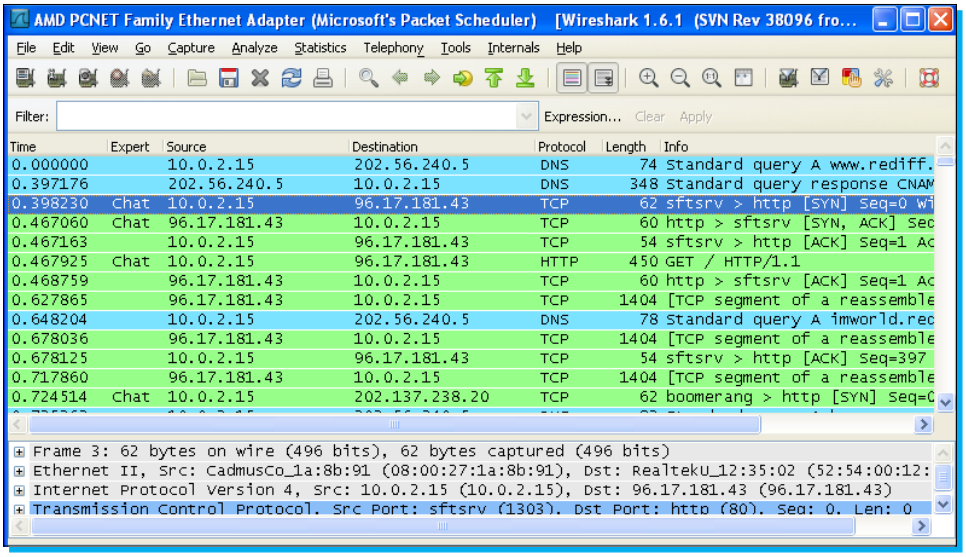
1. Select the **Columns** option from the left panel.

2. In the **properties** column (center bottom) click on the **Add** button. This will create a new column entry. From the **Field Type** drop-down menu, select **Expert Info Severity**.



3. Click on the **Add** button to enable this column view. You can rename the column by clicking on it and entering the name of your choice.

Once you are done with these three steps you will notice that an extra column with Expert Infos has been added in your capture panel.



This was a short description of using the preferences option to change the default view according to your choice. You can explore the other possibilities by changing the preferences of color, font, display protocols, and so on.

These were some of the kick-starter sections to get you started with Wireshark and packet capture. By now we have covered most of the basic and some advanced topics that are essential to understand the working of Wireshark. In the next section of this Starter guide we will cover some top features that can be very helpful for your packet analysis if you want to dive even deeper into the tool.

Top 5 features you need to know about

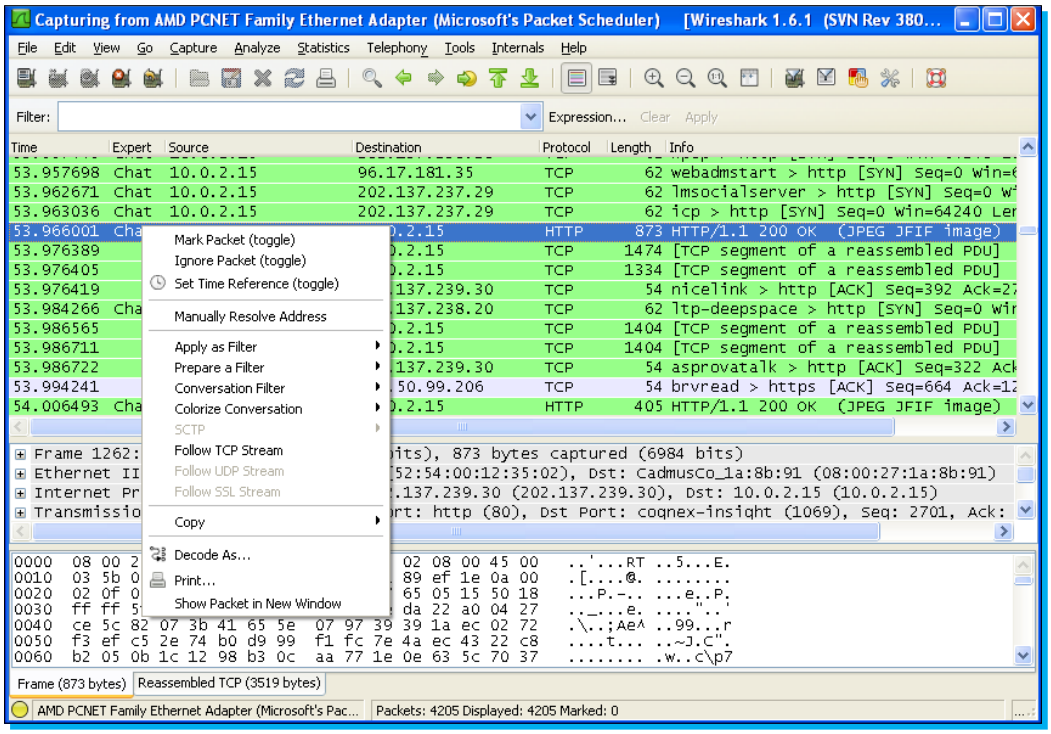
As you start to use Wireshark, you will realize that there are a wide variety of things that you can do with it. This section will teach you all about the most commonly performed tasks and most commonly used features in Wireshark.

Working with packet streams

While working on network capture, there can be multiple instances of network activities going on. Consider a small example where you are simultaneously browsing multiple websites through your browser. Several TCP data packets will be flowing across your network for all these multiple websites. So it becomes a bit tedious to track the data packets belonging to a particular stream or session. This is where **Follow TCP stream** comes into action.

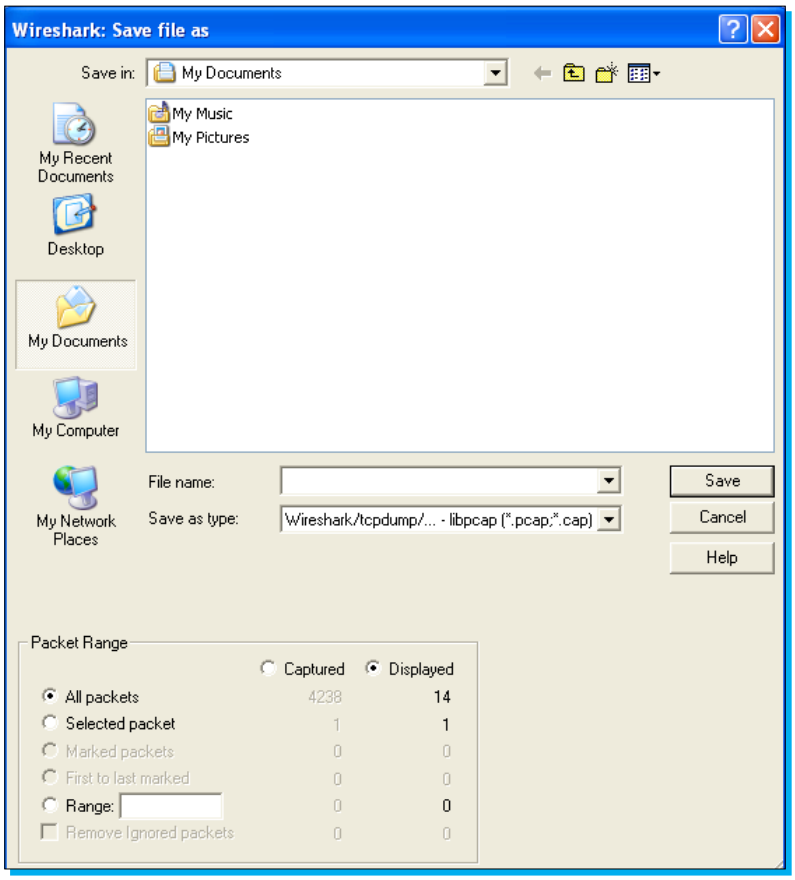
Now when you are visiting multiple websites, each site maintains its own stream of data packets. By using the **Follow TCP stream** option we can apply a filter that locates packets only specific to a particular stream.

To view the complete stream, select your preferred TCP packet (for example, a GET or POST request). Right-clicking on it will bring up the option **Follow TCP Stream**.

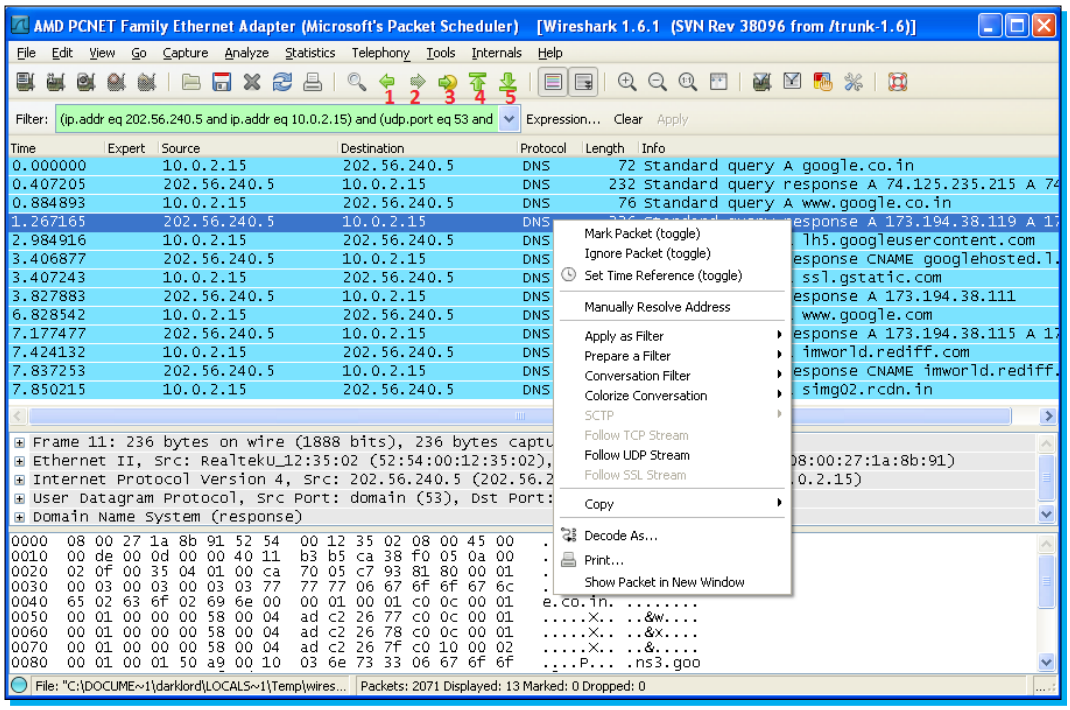


Once you click on **Follow TCP Stream**, you will notice that a new filter rule is applied to Wireshark and the main capture window reflects all those data packets that belong to that stream. This can be helpful in figuring out what different requests/responses have been generated through a particular session of network interaction. If you take a closer look at the filter rule applied once you follow a stream, you will see a rule similar to `tcp.stream eq <Number>`. Here `Number` reflects the stream number which has to be followed to get various data packets.

An additional operation that can be carried out here is to save the data packets belonging to a particular stream. Once you have followed a particular stream, go to **File | Save As**. Then select **Displayed** to save only the packets belonging to the viewed stream.



Similar to following the TCP stream, we also have the option to follow the UDP and SSL streams. The two options can be reached by selecting the particular protocol type (UDP or SSL) and right-clicking on it. The particular follow option will be highlighted according to the selected protocol.



The Wireshark menu icons also provide some quick navigation options to migrate through the captured packets. These icons include:

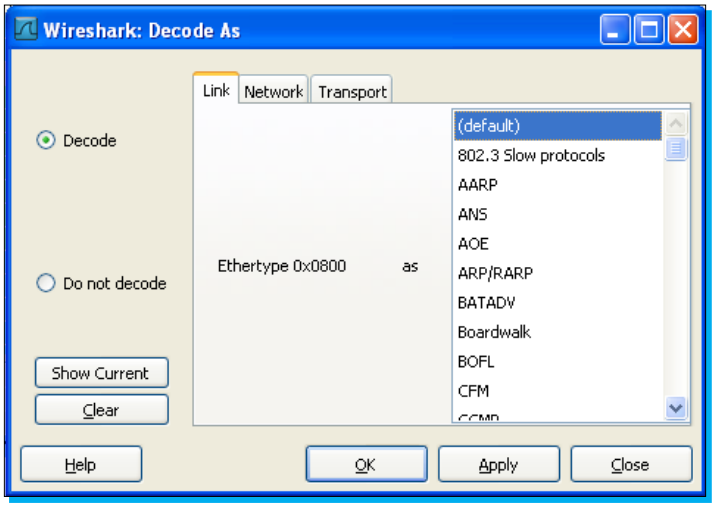
- **Go back in packet history (1)**: This option traces you back to the last analyzed/selected packet. Clicking on it multiple times keeps pushing you back to your selection history.
- **Go forward in packet history (2)**: This option pushes you forward in the series of packet analysis.
- **Go to packet with number (3)**: This option is useful in directly going to a specific packet number.
- **Go to the first packet (4)**: This option takes you to the first packet in your current display of the capture window.
- **Go to last packet (5)**: This option jumps your selection to the last packet in your capture window.

Decoding packets and exporting objects

Wireshark provides a very powerful feature of decoding the captured packets into user specified formats. The decode functionality of Wireshark temporarily diverts the specific protocol dissections. This process can be helpful in cases where the data is flowing over multiple ports or it is not recognized as a standard protocol. The decode functionality can be used to easily transform the packets into their original protocol.

The decode functionality is a temporary action and the changes will be lost unless the decoded packets are saved separately.

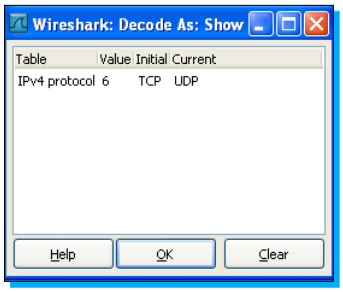
Right-clicking on a packet will reflect the option **Decode as...**. Clicking on it will open a box containing different tabs and related options.



You will notice three different tabs at the top. Let us understand them one by one.

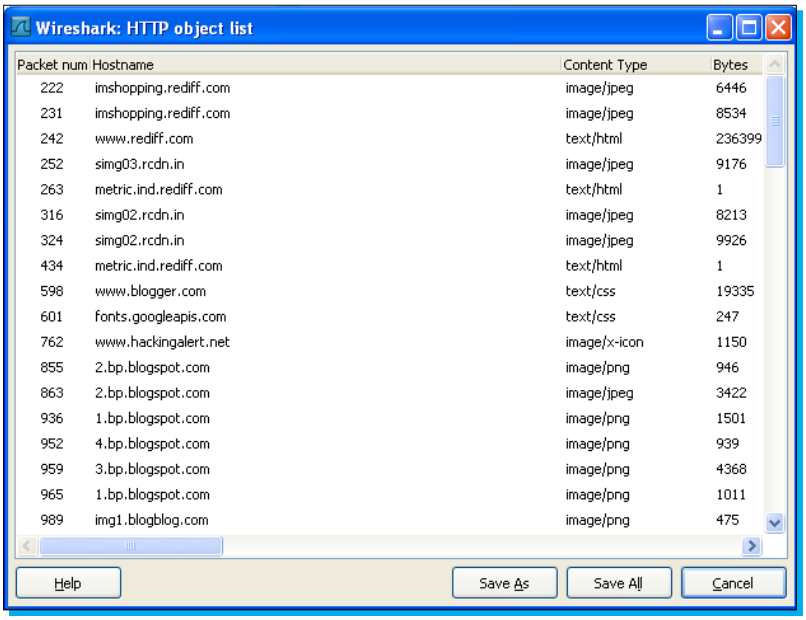
- **Decode As Link**: Selecting this tab will show different protocols at the data link layer. Choosing any of these protocols and selecting the **Decode** radio option will convert the specified packet into the user specified protocol.
- **Decode As Network**: This tab contains different network layer protocols such as TCP and UDP. It can be most useful in IPv4 to IPv6 conversion.
- **Decode As Transport**: This tab reflects the different transport layer protocols in which the data packet can be decoded.

You will also notice the **Show Current** button which shows the different user specified decodes.



Clicking on **Clear** will remove all the packet decodes applied by the user. Clicking on the **Apply** button applies the current decode option and keeps the dialog box open for further operation.

Moving ahead with another important discussion of this section, Wireshark also provides the feature to export different objects captured as packet streams. For example, downloading a flash will be seen in the form of packets in the capture window, but it can be exported as a standalone flash media by using the export feature. To view the different objects captured in an HTTP traffic, go to to **File** | **Export** | **Objects** | **HTTP**. This will open up a list of captured objects transferred as a part of HTTP traffic.



Selecting any of the object types and clicking on **Save As** will convert the packet capture into that object type. For example, selecting an image/JPEG file type and saving it will create a new image file on our local disk. This is a very efficient and quick way of retrieving different file types downloaded during packet capture.

Now let us move ahead with the next important feature of Wireshark that will be helpful in the quick analysis of data capture.

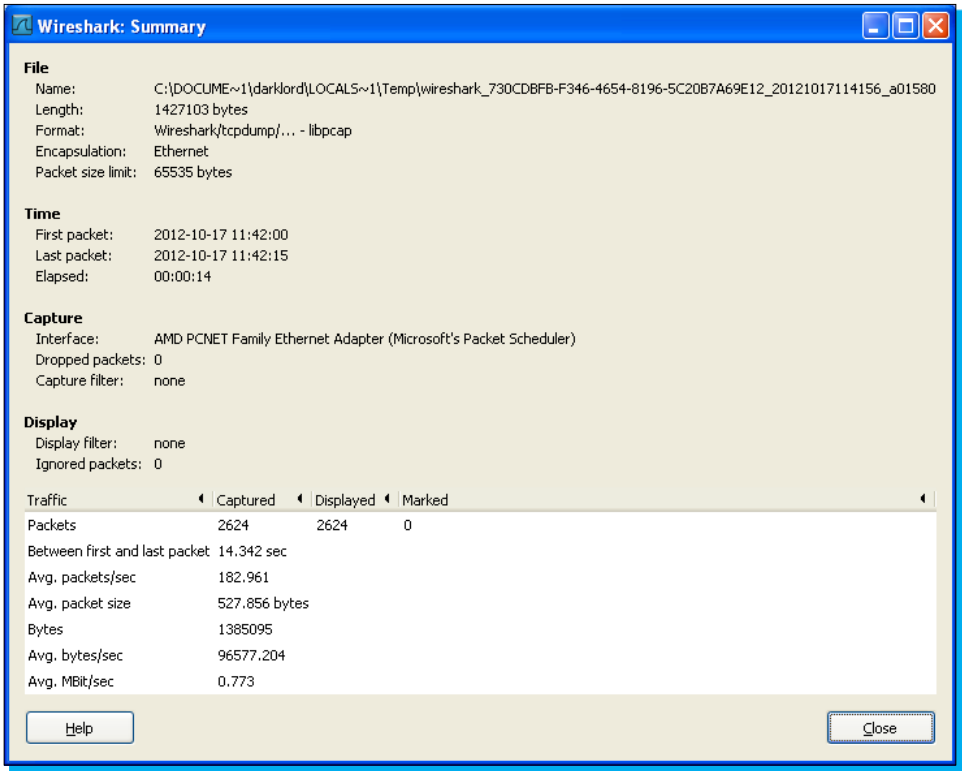
Statistics of the captured packets

In this section we will take a detailed look at generating statistics for the captured packets in order to have an overview of our network activity. We have already covered I/O graphs in our previous section. Here we will uncover some more important aspects for generating quick statistics of network capture.

The **Statistics** menu option of Wireshark provides some very cool statistical features that can be used to generate an overview of network activity. Here we will cover five important tools, namely Summary, Protocol Hierarchy, Conversations, Endpoint, and Flow Graph. Let us discuss each of them one by one.

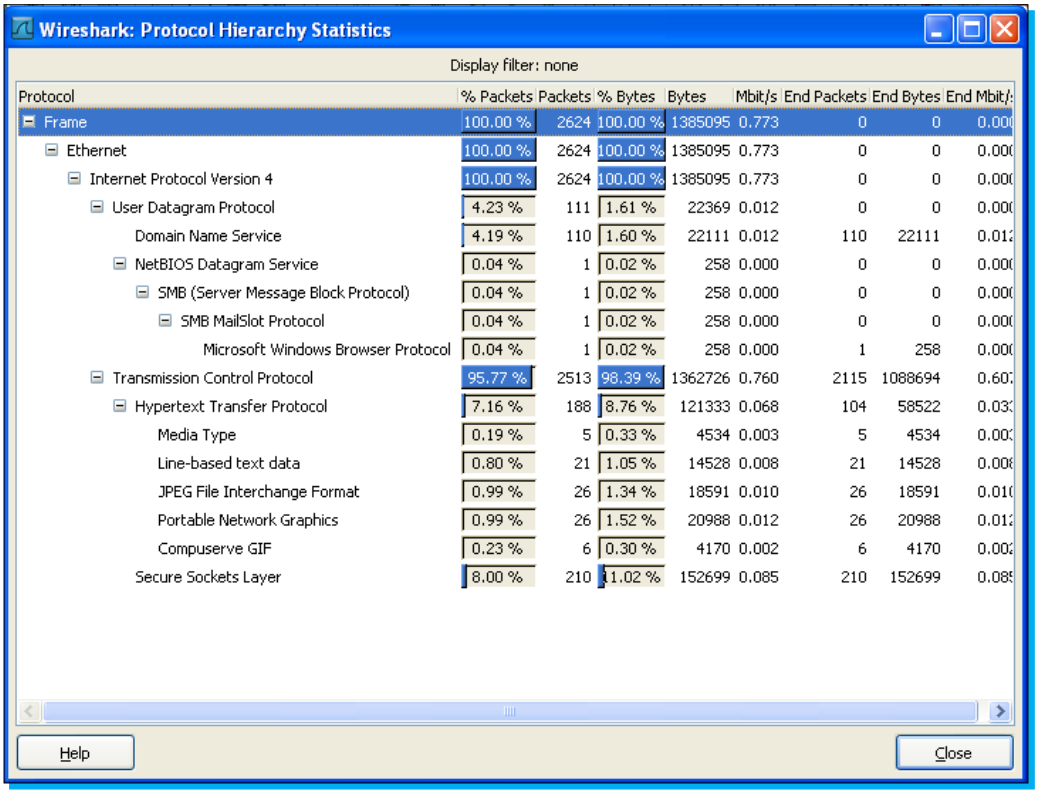
Summary

Summary returns a quick report about the entire capture process by reflecting key points such as interface, capture duration, and number and size of packets captured. It provides all the information in a nutshell. Summary is very useful during network forensics when a quick overview of network activity is required.



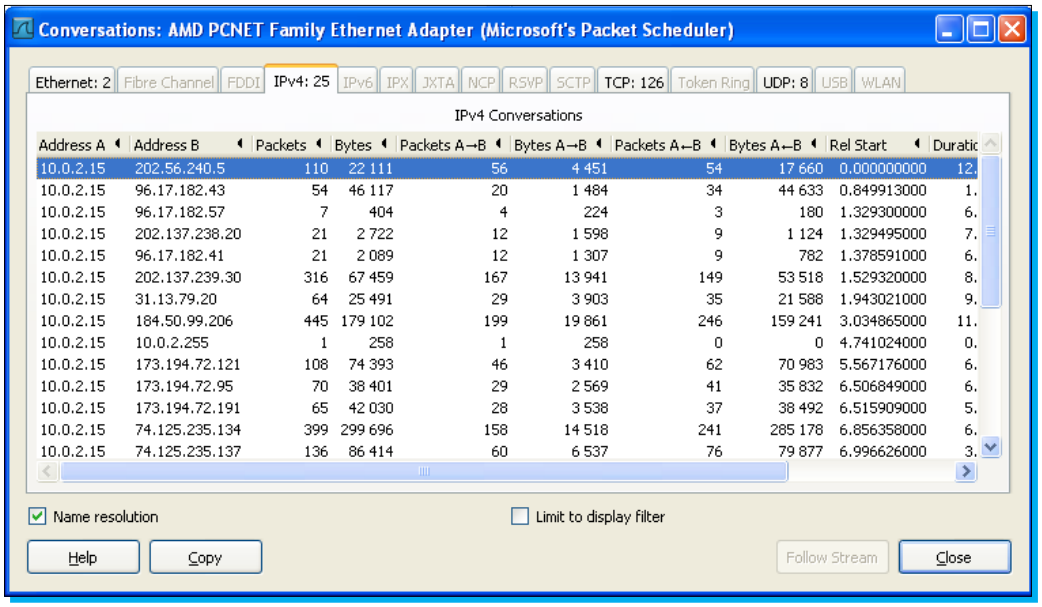
Protocol Hierarchy

Protocol Hierarchy reflects the statistical information of different protocols seen during network analysis in a nodal form. It arranges the protocols according to its layers and represents their existence in the captured file in percentage form.



Conversations

Conversation is another very interesting statistical analysis of data packets which shows information about communication between different IPs. It also shows Conversation statistics of different protocols transferred over an IP. This is the best statistical analysis when an end-to-end analysis has to be carried out between the host and server.

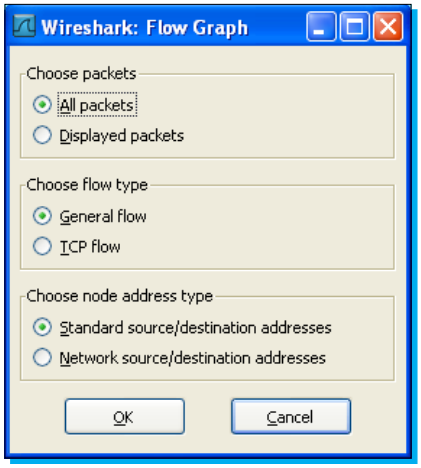


Endpoints

Endpoint statistics is similar to Conversations as it reflects the statistics of traffic to and from an IP address.

Flow graph

Flow graphs are a timeline-based representation of communication statistics. They represent the different conversations carried out based on time intervals. Clicking on **Flow Graph** launches a dialog box which asks for some initial information before generating the graph.



We can specify information such as choosing packets, flow type, and the mode in which the graph will reflect information. This statistical analysis is helpful in reading the timeline of network activity. It also helps in understanding the traffic load over time duration.

This was a basic demonstration of using different statistical options provided by Wireshark in order to have a better understating of our network capture. In the next section we will cover some technical aspects involved in Wireshark's reassembling and name resolution techniques.

Name resolution and packet reassembling

In this section we will look at some more important Wireshark features that make our network analysis process easier and more informative. We will start with name resolution.

Name resolution

Name resolution is the process of converting one identifying address into another for better understanding. For example, if a computer has a physical address of 00:16:CD:6C:8B:24, the DNS and ARP protocols allow us to view the readable name as MYPC-01. This association makes it easy to remember different devices on a network. Wireshark provides several name resolution tools that can convert physical addresses into corresponding readable device names. Let us take a quick look at some of the resolution tools present in Wireshark.

MAC name resolution

MAC name resolution uses the ARP/RARP protocol to convert a physical address (MAC) into its corresponding logical address (IP).

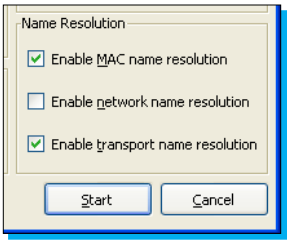
Network name resolution

Network name resolution further converts the logical address(IP) into its corresponding DNS name for better understanding. For example, 173.194.72.101 becomes `google.com`.

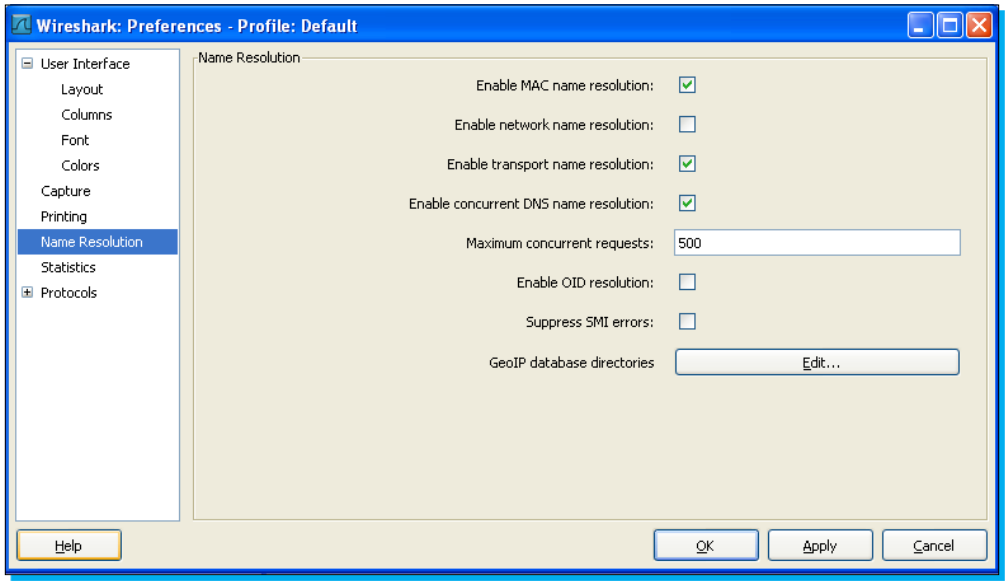
Transport name resolution

This name resolution technique identifies the service according to its port number. For example, port 80 hosts http service.

To enable name resolution in Wireshark, select **Capture | Capture Options**. This will launch the capture options dialog box which we have also seen earlier. The bottom-right side of this dialog box shows the **Name Resolution** checkbox for different categories.



You can select the different checkboxes based on your needs. Alternatively, you can also launch the **Name Resolution** dialog box by selecting **Edit | Preferences | Name Resolution**.



This provides even more advanced options as compared to the previous one. Here we can also provide a GeoIP database to Wireshark that can further improve the resolution power of Wireshark by helping it in identifying the packet regions based on its GeoIP.

This feature of Wireshark not only makes network analysis easier but also increases the capability of Wireshark in recognizing different protocols and services. But name resolution also has some drawbacks that need to be mentioned here.

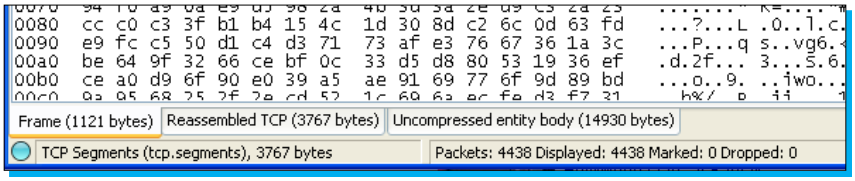
- Name resolution may fail during live capture or while opening a saved capture file
- Name resolution adds extra bytes of spaces into the packet file thus affecting the statistical graphs
- Name resolution adds additional overheads at packet processing

Now let us move to the next important discussion of this section.

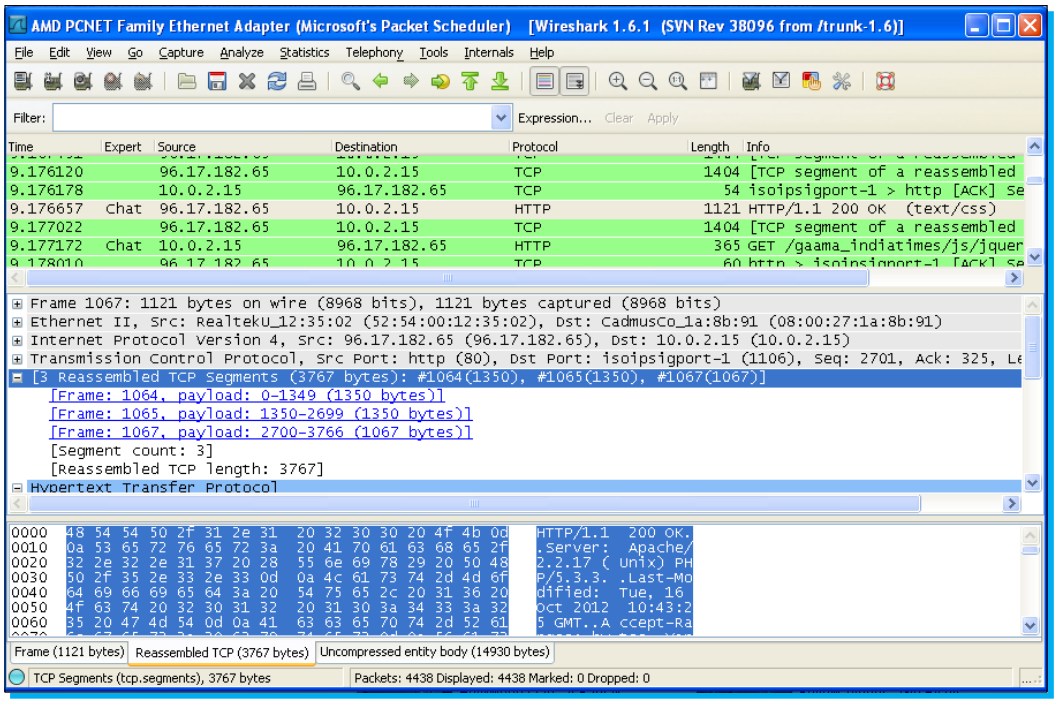
Packet reassembling

Packet reassembling is the process of transferring large chunks of data by splitting it into smaller packets and later on combining them again to form the complete data. In order to efficiently reproduce the captured data, Wireshark also implements its own packet reassembling technique.

Wireshark collects the different chunks of packets of the fragmented data and then reassembles it to reflect the complete file through the packet bytes panel.



For example, an HTTP GET request (HTML page) can be broken into smaller data packets. Once the transmission is complete, Wireshark reassembles the different packets and creates a single reassembled HTML page. The packet details panel adds extra information (**Reassembled TCP segments**) about the different packet numbers that are reassembled to form the complete set of data.



The complete reassembled data is reflected under the **Uncompressed entity body** tab of the packet bytes panel and its Hex dump is reflected under the **Reassembled TCP** tab.

Reassembling is enabled by default in Wireshark.

This was a quick discussion about how Wireshark manages large chunks of data packets and how it efficiently identifies different protocols, services, and devices. In the next and last section we will cover how to use Wireshark from the command line.

Wireshark command-line tools

Now that we have covered almost all basic implementations of Wireshark, we can finally conclude this section by covering Wireshark command-line tools, which are equally powerful like their GUI counterpart. Let us jump to analyze some of these commands on the Windows command-line. Similar commands can be used on the Linux terminal as well.

Tshark – terminal Wireshark

Tshark is the command to initiate most of the operations in Wireshark from the command line. To start with the command line on the Windows machine, locate the Wireshark installation directory which is by default `Program Files\wireshark`.

Passing the `tshark` help command will display a huge list of parameters that we can pass along with `tshark`. Let us execute this command and read its output.

```
C:\Program Files\Wireshark>tshark -h
TShark 1.6.1 (SVN Rev 38096 from /trunk-1.6)
Dump and analyze network traffic.
See http://www.wireshark.org for more information.

Copyright 1998-2011 Gerald Combs <gerald@wireshark.org> and
  contributors.
This is free software; see the source for copying conditions. There
  is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
  PURPOSE.

Usage: tshark [options] ...
```

Most of the parameter details are self-explanatory and you can easily understand their uses. Let us list the different interfaces available to us using `tshark`.

```
C:\Program Files\Wireshark>tshark -D
1. \Device\NPF_{730CDBFB-F346-4654-8196-5C20B7A69E12} (AMD PCNET
  Family Ethernet
 Adapter (Microsoft's Packet Scheduler) )
2. \Device\NPF_{DD763ED6-6670-4281-8551-A3C651241F95} (AMD PCNET
  Family Ethernet
 Adapter (Microsoft's Packet Scheduler) )
```

As you can see, the `-D` parameter has been passed to list the available interfaces. Similarly we can start capturing through any interface by using the `-i` parameter followed by the interface number and the output can be stored into a file by using the `-w` parameter. We can combine the different parameters under a single command to perform our packet capture. Let us take an example query with different parameters passed to `tshark`.

```
C:\Program Files\Wireshark>tshark -i 1 -w capture.pcap -p -c 100
Capturing on AMD PCNET Family Ethernet Adapter (Microsoft's Packet
  Scheduler)
100


C:\Program Files\Wireshark>
```

You can see the different parameters passed to the `tshark` command. The complete command can be explained in simple language as: capture from interface number `1`, save the output with filename `capture.pcap`, do not capture is promiscuous mode (-p), and stop capture (-c) when the packet count reaches `100`.

You can further combine other parameters along with this command according to your need. This command will create a file named `capture.pcap` in the present working directory, that is, `/program files/wireshark`.

The next step can be to view the captured file. To do this we can use the `-r` parameter followed by the input file name.

```
C:\Program Files\Wireshark>tshark -r capture.pcap


  0.000000     10.0.2.15 -> 173.194.72.94 TLSv1 91 Application Data
  0.000822  173.194.72.94 -> 10.0.2.15    TCP 60 https > opsmgr [ACK]
  Seq=1 Ack=
38 Win=65535 Len=0
  0.148540  173.194.72.94 -> 10.0.2.15    TLSv1 91 Application Data
  0.314759     10.0.2.15 -> 173.194.72.94 TCP 54 opsmgr > https [ACK]
  Seq=38 Ack
=38 Win=62839 Len=0
  0.527277     10.0.2.15 -> 31.13.79.20  TLSv1 606 Application Data
  0.527761   31.13.79.20 -> 10.0.2.15    TCP 60 https > hiq [ACK]
  Seq=1 Ack=553
Win=65535 Len=0
  1.097436   31.13.79.20 -> 10.0.2.15    TCP 1404 [TCP segment of a
  reassembled
PDU]
  1.107744   31.13.79.20 -> 10.0.2.15    TLSv1 229 Application Data
```

```
  1.107790     10.0.2.15 -> 31.13.79.20  TCP 54 hiq > https [ACK]
  Seq=553 Ack=15
26 Win=64240 Len=0
  1.127783   31.13.79.20 -> 10.0.2.15    TCP 1404 [TCP segment of a
  reassembled
PDU]
  1.147908   31.13.79.20 -> 10.0.2.15    TLSv1 1234 Application Data
  1.147942     10.0.2.15 -> 31.13.79.20  TCP 54 hiq > https [ACK]
  Seq=553 Ack=40
56 Win=64240 Len=0
  1.167471   31.13.79.20 -> 10.0.2.15    TCP 1404 [TCP segment of a
  reassembled
PDU]
```

You will notice an output similar to the GUI where information such as capture time, source/destination, and protocol is reflected. This is how we can leverage the power of Wireshark through the command line. Let us work on some more interesting command-line tools.

Rawshark – dumping and analyzing the traffic

Rawshark is used to read the stream of packets from a captured file. This operation is similar to **Follow TCP Stream** which we covered in the *Working with packet streams* section. To view the available options of Rawshark, we can use the `rawshark -h` command.

A simple Rawshark command will look like the following:

```
C:\Program Files\Wireshark>rawshark -d proto:http -r capture
```

This command looks for HTTP protocol within the specified filename.

editcap

`editcap` is used to apply additional operations on captured files such as propping packets, error correction, and filter. Its different options can be seen using the `editcap -h` command. A simple `editcap` command will look like the following:

```
C:\Program Files\Wireshark>editcap -d capture.pcap capture1.pcap 1
  100
Add_Selected: 1
Not inclusive ... 1
Add_Selected: 100
Not inclusive ... 100
100 packets seen, 0 packets skipped with duplicate window of 5
  packets.
```

The command drops duplicate packets in the range of 1 to 100 packets present in the `capture.pcap` file and stores the remaining packets under the `capture1.pcap` file.

mergecap

`mergecap` is used to merge two or more captured files into a single file. It provides the option of either concatenating or truncating the selected capture files. A simple `mergecap` command that will concatenate two `pcap` files is as follows:

```
C:\Program Files\Wireshark>mergecap -a -w newcapture.pcap
  capture.pcap capture1.pcap
```

text2pcap

`text2pcap` reads the ASCII hex dump file and writes the data described into a `libpcap` style capture file. `text2pcap` can read hex dumps with multiple packets in them, and build a capture file of multiple packets. An example command is as follows:

```
C:\Program Files\Wireshark>text2pcap -o h text.txt converted.pcap
Input from: text.txt
Output to: converted.pcap
Wrote packet of 37 bytes at 0
Wrote packet of 37 bytes at 37
Read 2 potential packets, wrote 2 packets
```

This was a simple demonstration of some of the important Wireshark command-line tools. You will notice that similar features are also present in the GUI mode.

With this we come to an end of this section. In the next section we will perform a live capture and implement various methodologies for packet analysis that we have seen so far in this guide.

Wireshark activity

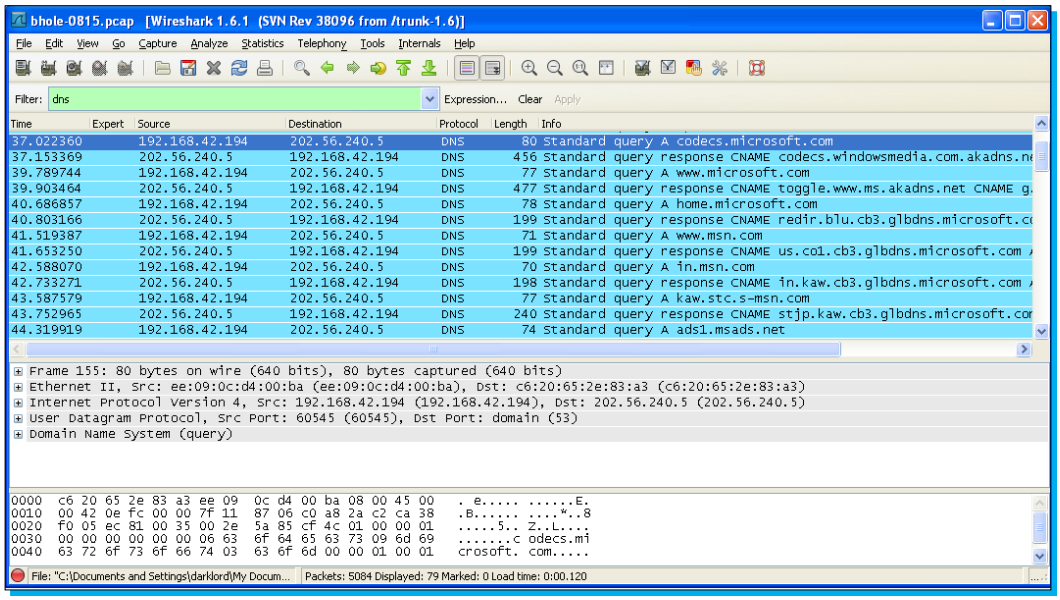
Now that we have covered almost all the basic features of Wireshark, in this section we will try and practically implement the techniques we have learnt so far in the previous sections. This exercise will be helpful in giving you a practical insight of various processes that are followed while performing network forensics. We will be taking an example where a user visits a website while browsing the Internet but after a few minutes he notices some weird behavior on his system. Fortunately, the user has captured a `pcap` file that can help us in analyzing what exactly happened on the wire.

For your better understanding, you can download the capture file from the following location and follow up with this section:

`https://www.dropbox.com/s/ykl6rhsfstuw0fp/wireshark_starter.pcap`

Once you have opened the packet capture in Wireshark, the first step will be to look at the different DNS queries that are made while the user was online. This will give us a quick view of any malicious DNS request that may have been made without the knowledge of the user. This can happen if the website that the user has visited contains a hidden iframe of some malicious script inside it.

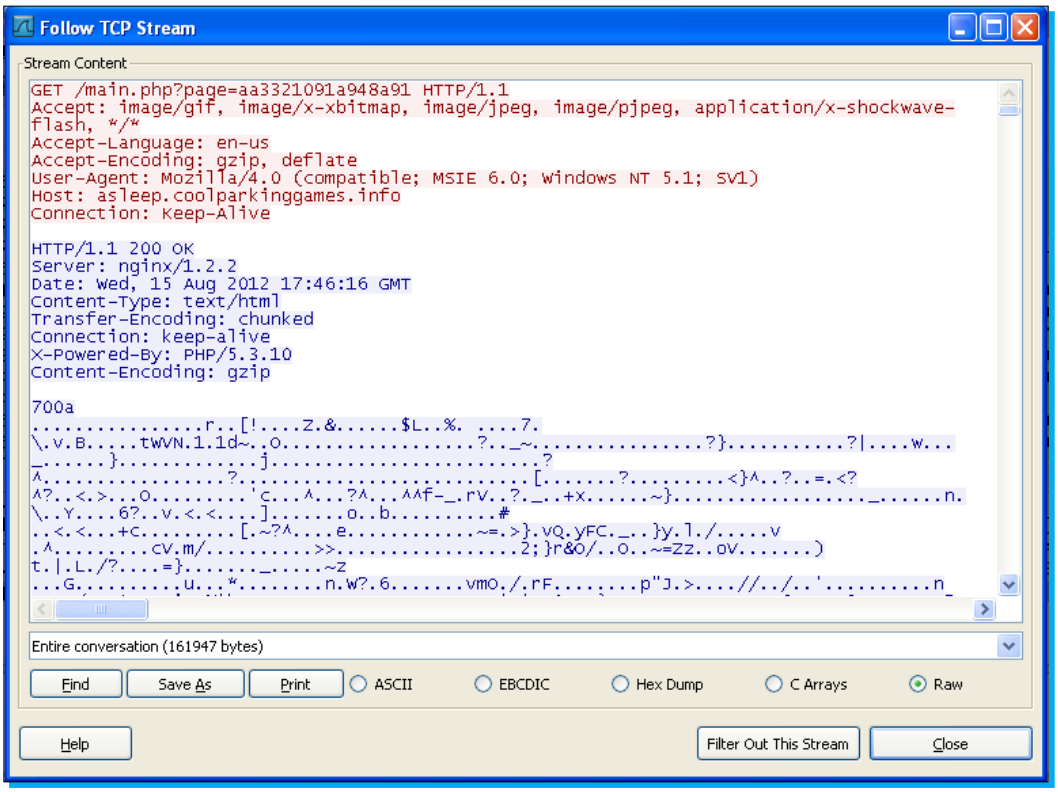
Typing `dns` in the **Filter** box and pressing *Enter* will return different DNS resolutions that occurred over the wire.



You will notice lots of DNS name resolutions being carried out. It looks a bit difficult to figure out if there is anything suspicious in it. The next thing that you can do is to look for different HTTP streams that have flown during the network activity. Typing `http` and pressing *Enter* will return different HTTP packets. You will notice several HTML, JavaScript, and image traffic become visible to you. Taking a closer look at it, you will also find some 302 redirections, non-HTTP streams, and Java Archive downloads. This can raise some suspicion in the investigators' minds that the system may have been infected using a Java exploit. To investigate further, you can look at different GET requests made through the machine. Typing `tcp contains "GET"` will reflect all the GET requests in the capture panel.

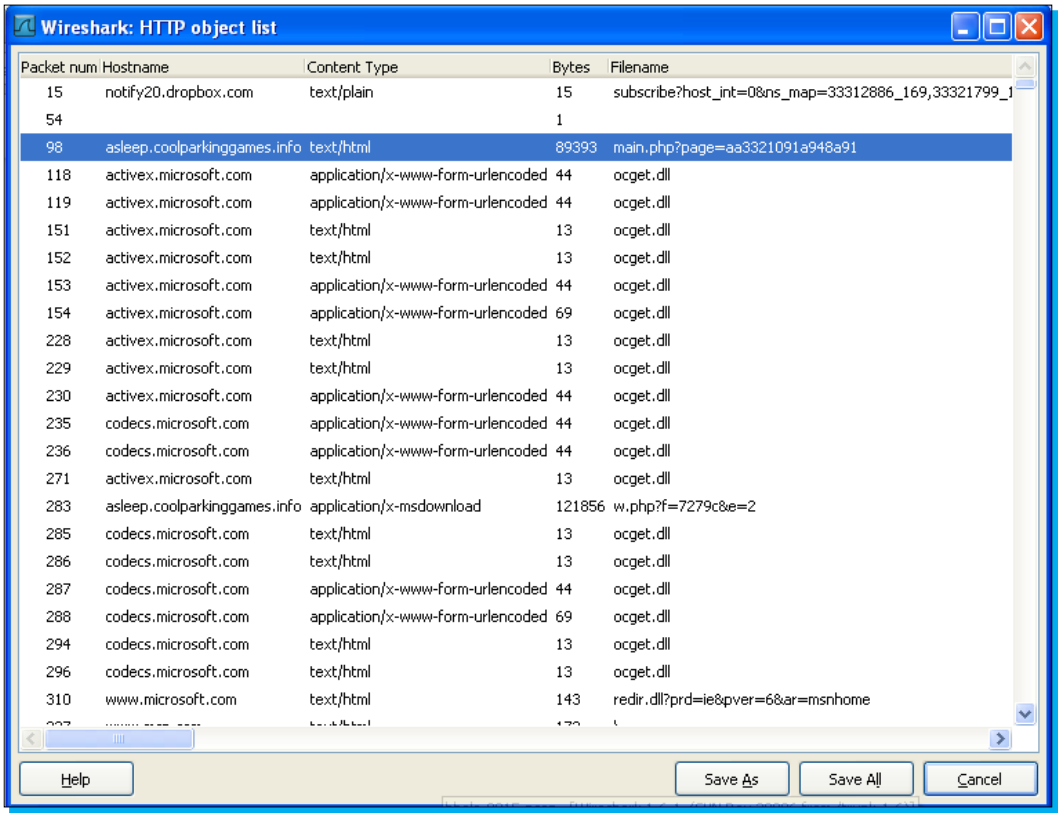
You will notice a `GET` request with an unusual URL pattern of `?page=<random numbers>`.

You can right-click on it and follow the stream to see the original source code of the GET request.



You will notice that the page content is gzip encoded. Now to view the source code in plain text, you will have to export is as an HTML page and then view its source. To do this, we can go to **File | Export | Objects | HTTP**.

This will open the dialog box that contains the complete list of different HTTP objects present in the packet streams.



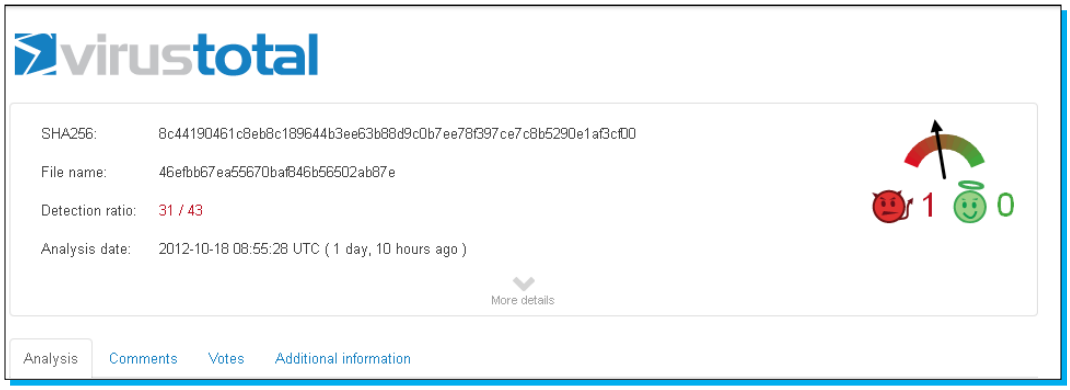
Now you can select the HTML page and click on **Save As** to export it onto your disk. Then the webpage can be opened in any HTML editor to view its source code. Taking a closer look at the **Export** dialog box, you will notice that the application `/x-msdownload executable file(exe)` is also being downloaded from the URL pattern `w.php?f=<random>`. This can be a dangerous executable that might have infected the user's system. You can select the object and export it as an executable. Alternatively, you can export all the objects by selecting **Save All**.

Coming back to our mysterious HTML page, once it has been exported, you can view its original source by opening it in any text editor. You will notice lots of random numbers and alphabets. Scrolling down to the bottom of the page will show some JavaScript obfuscation that is used as the decoding algorithm for those long random texts. A quick search of some of the lines of JavaScript on Google reveals that they are the parts of the well known Blackhole exploit kit.

```
d=11;
for(i=33920-1;i!=-1;i--){
    w=i;
    v=a[w];
    dd=33920-i-2+1;
    b=d;
    dd=dd-b*eval("Ma"+"th").floor(dd/d);
    k=v*1-(dd-13);
    kk=k;
    c+=String.fromCharCode(kk);
}
if(020==0x10)eval(c);}
if(window.document)for(i=0;i<document.body.childNodes.length;i++){
    if (document.body.childNodes[i].getAttribute("data")!=null){
        if(z)g=document.body.childNodes[i].innerHTML.replace(/hwehea/g,"");
    }
}
a=[];
for(i=0;i<g.length;i+=2){
    gg=g.substr(i,0x2);
    a.push(parseInt(gg,0x10));
```

Blackhole is a web-based exploit kit that compromises the user's system by exploiting browser vulnerabilities in applications such as Java, PDF, and ActiveX. Once the user system has been exploited, the exploit kit downloads malwares and viruses and infects the system. If you remember, the application file that we exported can be a malicious malware or virus that would have been downloaded once the user's browser was exploited by the exploit kit. To quickly analyze the sample, it can be uploaded to malware analysis websites such as `virustotal.com` or can be scanned by an effective antivirus product.

Uploading the file on `virustotal.com` shows that 33 out of 43 antivirus products detect it as a virus.



This finally proves that the user's system was exploited by the Blackhole exploit kit which dropped a virus that resulted into the malfunctioning of the machine.

This was a short demonstration of how Wireshark can be used in network forensics to identify malicious activities over the network. The complexity of the analysis may vary based on the situation but the underlining principle remains the same.

In the last section of this book we will look into some additional information that can be helpful for you in carrying your Wireshark study forward.

People and places you should get to know

If you need help with Wireshark, here are some people and places which will prove invaluable.

Official sites

- Home page: `http://www.wireshark.org/`
- Manual and documentation: `http://www.wireshark.org/docs/`
- Wiki: `http://wiki.wireshark.org/`
- Blog: `http://blog.wireshark.org/`
- Source code :`http://anonsvn.wireshark.org/viewvc/`

Articles and tutorials

- *How to Use Wireshark to Capture, Filter, and Inspect Packets*: `http://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/`
- *5 Killer Tricks to Get the Most Out of Wireshark*: `http://www.howtogeek.com/106191/5-killer-tricks-to-get-the-most-out-of-wireshark/`
- *Network Analysis with Wireshark On Ubuntu 9.10*: `http://www.howtoforge.com/network-analysis-with-wireshark-on-ubuntu-9.10`
- *Packet Sniffin' Tips and Tricks that I wished I would have known earlier in my career...*: `https://learningnetwork.cisco.com/blogs/network-sheriff/2008/08/25/packet-sniffin-tips-and-tricks-that-i-wished-i-would-have-known-earlier-in-my-career`

Community

- Official mailing list: `https://www.wireshark.org/mailman/listinfo/wireshark-dev`
- Unofficial forums: `http://www.techsupportforum.com/forums/f139/ethereal-wireshark-131427.html`
- Official IRC channel: `irc://irc.freenode.net/wireshark`
- User FAQ: `http://www.wireshark.org/faq.html`

Blogs

- Wireshark official blog: `http://blog.wireshark.com/`
- The Cisco blog is a very useful source for getting the latest information on network sniffing: `https://learningnetwork.cisco.com/`
- The tech republic blog is also an important source to learn about Wireshark: `http://www.techrepublic.com/blog/opensource/`

Twitter

- Riverbed Technology: `https://twitter.com/riverbed`
- Abhinav Singh: `https://twitter.com/abhinavbom`
- For more open source information, follow Packt at `http://twitter.com/#!/packtopensource`

Thank you for buying
Instant Wireshark Starter

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

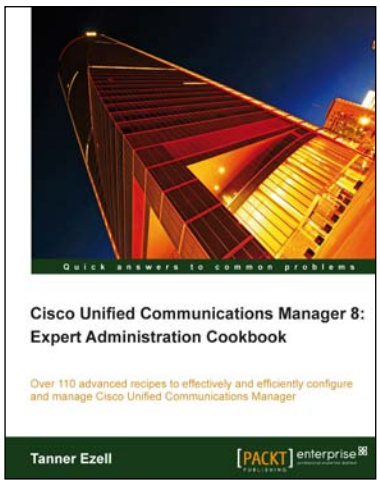
Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] PUBLISHING

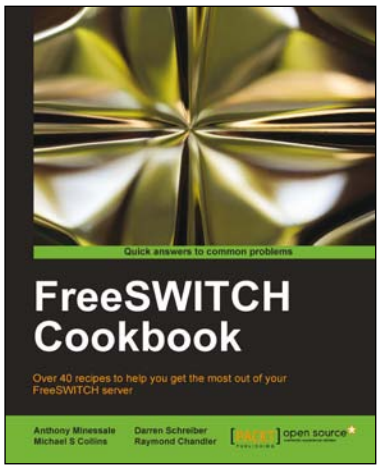


Cisco Unified Communications Manager 8: Expert Administration Cookbook

ISBN: 978-1-84968-432-3 Paperback: 310 pages

Over 110 advanced recipes to effectively and efficiently configure and manage Cisco Unified Communication Manager

1. Full of illustrations, diagrams, and tips with clear step-by-step instructions and real time examples
2. Master call admission control and the technologies associated with it, which is an important aspect of any unified communications deployment to ensure call quality and resilience.
3. Master the necessary aspects of configuring media resources to support conferencing, transcoding, and Music on Hold.



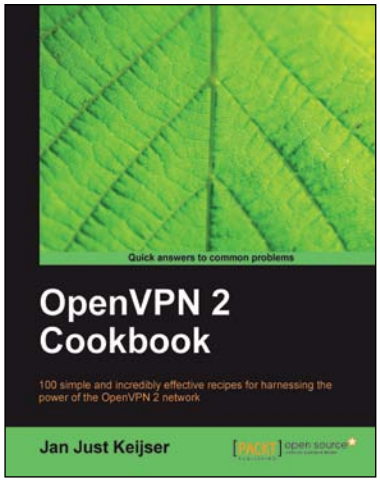
FreeSWITCH Cookbook

ISBN: 978-1-84951-540-5 Paperback: 150 pages

Over 40 recipes to help you get the most out of your FreeSWITCH server

1. Get powerful FreeSWITCH features to work for you
2. Route calls and handle call detailing records
3. Written by members of the FreeSWITCH development team

[PACKT]
PUBLISHING

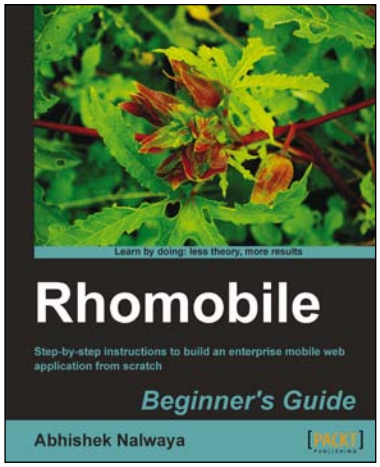


OpenVPN 2 Cookbook

ISBN: 978-1-84951-010-3 Paperback: 356 pages

100 simple and incredibly effective recipes for harnessing the power of the OpenVPN 2 network

1. Set of recipes covering the whole range of tasks for working with OpenVPN
2. The quickest way to solve your OpenVPN problems!
3. Set up, configure, troubleshoot and tune OpenVPN



Rhomobile Beginner's Guide

ISBN: 978-1-84951-516-0 Paperback: 306 pages

Step-by-step instructions to build an enterprise mobile web application from scratch

1. Explore all of Rhomobile's features and products through the creation of a mobile web application
2. Step-by-step instructions help you to build an enterprise mobile web application from scratch, through to deployment
3. Clear guides for developing applications on iPhone, Blackberry, and other smartphones