Wireshark Network Security

Wireshark is the world’s foremost network protocol analyzer for network analysis and troubleshooting.

This book will walk you through exploring and harnessing the vast potential of Wireshark, the world’s foremost network protocol analyzer.

The book begins by introducing you to the foundations of Wireshark and showing you how to browse the numerous features it provides. You’ll be walked through using these features to detect and analyze the different types of attacks that can occur on a network. As you progress through the chapters of this book, you’ll learn to perform sniffing on a network, analyze clear-text traffic on the wire, recognize botnet threats, and analyze Layer 2 and Layer 3 attacks along with other common hacks.

By the end of this book, you will be able to fully utilize the features of Wireshark that will help you securely administer your network.

Who this book is written for

If you are network administrator or a security analyst with an interest in using Wireshark for security analysis, then this is the book for you. Basic familiarity with common network and application services terms and technologies is assumed.

What you will learn from this book

- Familiarize yourself with the robust features offered by Wireshark
- Use the powerful command-line utilities shipped with Wireshark
- Analyze numerous threats to network security using Wireshark
- Investigate attacks performed using popular security tools such as Nmap, Nessus, Metasploit, and more
- Solve real-world CTF challenges using Wireshark
- Create your own security-related profile in Wireshark
- Configure Wireshark for effective network troubleshooting
- Get accustomed to common scenarios faced by security analysts
- Analyze malware traffic successfully by using Wireshark
- Unearth anomalies hampering the speed of network communications

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 2 'Tweaking Wireshark'
- A synopsis of the book’s content
- More information on Wireshark Network Security
About the Author

**Piyush Verma** currently serves as a senior security analyst at NII Consulting, India, and enjoys hacking his way into organizations (legally) and fixing the vulnerabilities encountered. He strongly values hands-on experience over certifications; however, here are a few certifications he has earned so far: OSCP, CEH, CHFI, CCNA Security, and CompTIA Security+. He is a highly sought-after professional speaker and has delivered security training to folks working in public, private, and "secret" sectors. He can be contacted at [https://in.linkedin.com/in/infosecpiyushverma](https://in.linkedin.com/in/infosecpiyushverma).
Preface

Wireshark is the tool of choice for network administration and troubleshooting, but its scalability goes beyond that. It is an excellent aid in performing an in-depth analysis of issues pertaining to the overall security of the network. Several tools and devices are available in the market to detect network-related attacks and take appropriate actions based on a predefined set of rules. However, at a very granular level, it all boils down to frames, or sometimes interchangeably called as packets, and the data they carry.

This book is written from the standpoint of using Wireshark to detect security-concerning flaws in commonly used network protocols and analyze the attacks from popular tools such as Nmap, Nessus, Ettercap, Metasploit, THC Hydra, and Sqlmap. In the later part of the book, we will dive into inspecting malware traffic from an exploit kit and IRC botnet and solve real-world Capture-The-Flag (CTF) challenges using Wireshark, basic Python code, and tools that complement Wireshark.

What this book covers

Chapter 1, Getting Started with Wireshark – What, Why, and How?, provides an introduction to sniffing and packet analysis and its purpose. Later, we will look at where Wireshark fits into the picture and how it can be used for packet analysis by performing our first packet capture.

Chapter 2, Tweaking Wireshark, discusses the robust features of Wireshark and how they can be useful in terms of network security. We will briefly discuss the different command-line utilities that ship with Wireshark.
Preface

Chapter 3, *Analyzing Threats to LAN Security*, dives into performing sniffing and capturing user credentials, analyzing network scanning attempts, and identifying password-cracking activities. In this chapter, we will also learn to use important display filters based on protocols and common attack-tool signatures and also explore regular expression-based filters. Then we will look at tools that complement Wireshark to perform further analysis and finally nail an interesting CTF challenge via the techniques learned in the chapter.

Chapter 4, *Probing E-mail Communications*, focuses on analyzing attacks on protocols used in e-mail communication and solving a couple of real-world e-mail communication challenges using Wireshark.

Chapter 5, *Inspecting Malware Traffic*, starts with creating a new profile under Wireshark for malware analysis and then picks up a capture file from an exploit kit in action and diagnoses it with the help of Wireshark. Later, we also give a brief on inspecting IRC-based botnets.

Chapter 6, *Network Performance Analysis*, begins by creating a troubleshooting profile under Wireshark and then discusses and analyzes TCP-based issues and takes up case studies of slow Internet, sluggish downloads, and delves further into picking up on Denial-of-Service attacks using Wireshark.
It goes without saying that once you start sniffing on a busy network, you will be flooded with a bulk load of traffic, and in no time you may lose track of what you were looking for and seek assistance. Therefore, it becomes vital to understand the different features that come with the sniffer. This chapter will focus on such features while analyzing multiple trace files using Wireshark. At the end of this chapter, you will be comfortable with:

- Working with filters in Wireshark
- Creating multiple profiles
- Using advanced techniques
- Performing command-line fu with handy utilities that come prepackaged with Wireshark

**Filtering our way through Wireshark**

Filters are like conditionals that programmers/developers use while writing code. If we only wanted to see the ARP packets in the `TelnetCapture.pcap` file, we will apply a condition in the **Filter** toolbar for ARP and if the current file contains ARP packets, they will be displayed else no packets will be seen at all.

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Time to live</th>
<th>Info</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>VMware_be:bf:94</td>
<td>Broadcast</td>
<td>ARP</td>
<td>42</td>
<td></td>
<td>who has 192.168.20.137</td>
<td>Telnet 192.168.20.138</td>
</tr>
<tr>
<td>2</td>
<td>0.000036</td>
<td>VMware_39:12:92</td>
<td>VMware_be:bf:94</td>
<td>ARP</td>
<td>60</td>
<td></td>
<td>192.168.20.137 is at 00:0c:29:39:12:92</td>
<td>Telnet 192.168.20.138</td>
</tr>
<tr>
<td>6</td>
<td>0.175392</td>
<td>VMware_39:12:92</td>
<td>Broadcast</td>
<td>ARP</td>
<td>60</td>
<td></td>
<td>who has 192.168.20.27</td>
<td>Telnet 192.168.20.137</td>
</tr>
<tr>
<td>7</td>
<td>0.000271</td>
<td>VMware_e9:alic8</td>
<td>VMware_39:12:92</td>
<td>ARP</td>
<td>60</td>
<td></td>
<td>192.168.20.2 is at 00:50:4e:9:alic8</td>
<td>Telnet 192.168.20.137</td>
</tr>
</tbody>
</table>

Only ARP traffic on display.
The current stable version, 1.12.6, of Wireshark includes a total 13 default capture filters and 15 default display filters. To look at the list of available capture filters, we can go to the Menu bar, click on **Capture | Capture Filters...**, and to look at the available display filters, click on the **Filter** button on the **Filter** toolbar. We can use these as is, or we can use them as templates and customize them to add/create new ones to suit our needs.

![Wireshark: Capture Filter - Profile: Default](image)

Default capture filters available in Wireshark

Wireshark provides the following two types of filtering options:

- Capture filters
- Display filters
The syntax for capture and display filters is different. Capture filters use Berkeley Packet Filtering (BPF) filter syntax also used by tcpdump, whereas display filters use Wireshark's specialized display filter format. To explore these filters in depth, please visit the following URLs:
Capture filters: http://wiki.wireshark.org/CaptureFilters
Display filters: http://wiki.wireshark.org/DisplayFilters

Capture filters
Capture filters are used before starting the capture on any interface and cannot be applied to an existing capture file.

When we know exactly what we're looking for, there is nothing better than capture filters. For example, when we need to troubleshoot Dynamic Host Configuration Protocol (DHCP)-related issues on a network and are not concerned with any other frames on the network, then we can apply the following capture filter: `port bootpc`, and all we will see is the DHCP traffic over the wire and nothing else.

Technically, all the traffic passes through the capture filter first and is then forwarded to the capture engine for further processing. In case a capture filter is applied, the frames that match the condition (capture filter) will be forwarded to Wireshark's capture engine and the rest will be completely discarded. This is the primary benefit of using capture filters as it offloads the computer from having to parse any useless frames. But this is a double-edged sword and we need to be careful when applying capture filters because we don't want to drop any frames that might be important from an analysis perspective.

Possessing an excellent set of capture filters in the arsenal can help us quickly pinpoint any anomaly on the network.

Another important point to be noted with respect to quick resolution of network issues is placing the analyzer at the right place, that is, location. As an example, if a lot of clients on the network complain about the network performance, then placing the analyzer closer to the server will be a good place to start, rather than analyzing at every client.
The following is a list of the capture filters:

Apart from the default set of the capture filters mentioned earlier, there are a number of capture filters that are handy to have in your arsenal. They are as follows:

<table>
<thead>
<tr>
<th>Capture filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ether host &lt;Client's MAC&gt; and ether host &lt;Server's MAC&gt;</td>
<td>Client-and-server only traffic, based on their respective MAC addresses</td>
</tr>
<tr>
<td>port bootpc</td>
<td>DHCP only traffic</td>
</tr>
<tr>
<td>vlan &lt;vlan-id&gt;</td>
<td>For a specific VLAN</td>
</tr>
<tr>
<td>ip6</td>
<td>IPv6 only traffic</td>
</tr>
<tr>
<td>ip proto 1</td>
<td>ICMP only traffic</td>
</tr>
<tr>
<td>port ftp</td>
<td>FTP only traffic</td>
</tr>
<tr>
<td>not port 3389</td>
<td>Exclude RDP traffic</td>
</tr>
<tr>
<td>udp dst port 162</td>
<td>SNMP requests</td>
</tr>
</tbody>
</table>

The useful link to generate capture filters is [https://www.wireshark.org/tools/string-cf.html](https://www.wireshark.org/tools/string-cf.html).

Whenever you're ambiguous about which capture filter to use, it is advisable to start off with a capture filter that is not too strict, or not use one at all and then narrow down the issue using display filters along the way. An example could be the use of the capture filter `udp dst port 162`, along with the display filter: `snmp.community`, to look at the community strings in the SNMP requests.

Display filters

Display filters are majorly used during analysis of already captured packets. However, they can also be used while capturing as they do not limit the packets being captured, they just restrict the visible number of packets.

Now, there will be times when we do not want to apply any filters before starting packet capture and need to capture everything that traverses our network.
For example, whenever a security incident is triggered on the network, it is important that we capture all the packets flowing on the wire and then analyze and reconstruct the event, using a packet/network analyzer tool such as Wireshark. During analysis, we might need to filter out traffic based on certain conditions, such as IRC-based communications or tracking down an FTP upload to a server in a different country. For the purpose of this, Wireshark provides **display filters** which makes life easier. Display filters allow us to take the maximum advantage of the Wireshark dissectors which take care of decoding and interpreting the fields of each packet.

There are tons of display filters available in Wireshark and memorizing them is not what we're supposed to do, luckily. In case we happen to know the field name, we can click on **Expressions** in the **Filter** toolbar and manually create one by selecting the **Field name** from the protocol subtree, the relation between the **Field name** and **Field value**, and then finally giving it a value.

Another way is to simply select the specific packet, locate the field we're looking for in the **Packet Details** pane, and the respective **Field name** for the filter will be highlighted in the Status bar at the bottom.
Tweaking Wireshark

As an example, we can see the following screenshot in which we are trying to find the **Field name** to use for filtering traffic based on TCP source port of 23.

The final filter is shown as follows:

```
Filter: tcp.srcport == 23
```

Display filter for source port -23 [TCP]

### The list of display filters

The following table shows a handy set of display filters:

<table>
<thead>
<tr>
<th>Display filter field names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip.addr</td>
<td>Traffic to or from an IP address</td>
</tr>
<tr>
<td>eth.addr</td>
<td>Traffic to or from an Ethernet address</td>
</tr>
<tr>
<td>tcp.port</td>
<td>Specify a TCP port</td>
</tr>
<tr>
<td>frame.time_delta</td>
<td>Time delta from the previous captured frame</td>
</tr>
<tr>
<td>http.request</td>
<td>HTTP requests only</td>
</tr>
<tr>
<td>arp.src.proto_ipv4</td>
<td>Sender IP in ARP packets</td>
</tr>
<tr>
<td>tcp.analysis.ack_rtt</td>
<td>Round-trip time</td>
</tr>
<tr>
<td>tcp.analysis.retransmission</td>
<td>Display all the retransmissions</td>
</tr>
</tbody>
</table>
### Display filter field names

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icmp.type</td>
<td>Type of ICMP packet</td>
</tr>
<tr>
<td>wlan.addr</td>
<td>Hardware address [Ethernet or other MAC address]</td>
</tr>
</tbody>
</table>

For a more comprehensive list of display filters, you can refer to the following links:

- [https://www.wireshark.org/docs/dfref/](https://www.wireshark.org/docs/dfref/)
- [http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf](http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf)

### Wireshark profiles

As we get comfortable using Wireshark, we will be creating several filters along the way, and some of them will be pretty neat and useful in critical situations. Also, there will be situations when fixing a particular issue requires the use of multiple display and/or capture filters, various colorization schemes to highlight bad/unexpected frames in the traffic assisting in visual distinction of such traffic, and customized preferences setting and layout changes. Therefore, creating our own profile for an attack scenario, a troubleshooting or any specific case is always a good option.

To look at the currently used profile in Wireshark, look at the bottom-right corner of the Status bar. So far, we have worked with the Default profile.

### Creating a new profile

To create a new profile, press a combination of Ctrl + Shift + A on the keyboard and click on New, or go to Edit | Configuration Profiles. We can also right-click on the Profile area in the Status bar, and select New.
The following screenshot shows multiple profiles created for different scenarios, plus the **Switch To** option, which makes it easy to switch between multiple profiles swiftly:

![Screenshot of Wireshark profile management]

Newly created profiles are highlighted.

Each profile configuration is located in different folders locally. To find the folder's location, simply go to **Help | About Wireshark** and select the **Folders** tab.

<table>
<thead>
<tr>
<th>Name</th>
<th>Folder</th>
<th>Typical Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td>C:\Users\Piyush Verma\Local Temp</td>
<td>capture files</td>
</tr>
<tr>
<td>Personal configuration</td>
<td>C:\Users\Piyush Verma\AppData\Roaming\Wireshark</td>
<td>untitled capture files</td>
</tr>
<tr>
<td>System</td>
<td>C:\Program Files\Wireshark</td>
<td>“ethers”, “ipxnet”</td>
</tr>
<tr>
<td>Program</td>
<td>C:\Program Files\Wireshark</td>
<td>program files</td>
</tr>
<tr>
<td>Personal Plugins</td>
<td>C:\Users\Piyush Verma\AppData\Roaming\Wireshark\plugins</td>
<td>dissector plugins</td>
</tr>
<tr>
<td>Global Plugins</td>
<td>C:\Program Files\Wireshark\plugins\1.12.3</td>
<td>dissector plugins</td>
</tr>
</tbody>
</table>

In the preceding screenshot, the highlighted portion contains the folder location for personal and global profiles.

In order to use your customized profile on another system, simply copy and paste the entire **profiles** folder to the other system's **profiles** folder.
Essential techniques in Wireshark

The techniques introduced under this section will provide you with the basic knowledge of what you will be dealing with, before diving deep into the packet analysis; these techniques are essential to understand from the packet analysis perspective. These mostly fall under the Statistics menu under the Menu bar as shown in the following figure:

Numerous options under the Statistics category

The Summary window

To access the Summary window in Wireshark, go to Statistics in the Menu bar and select Summary. The Summary window includes the following:

- File details
- Time details
- Capture details
- Display details
Important details that can be deduced from here are:

- Capture time and duration
- Version details of operating system and Wireshark
- Capture interface
- Any capture/display filter used
- Average packets/sec, average packet size
- Average bytes/sec

The Protocol Hierarchy window

To view this, go to Statistics in the Menu bar and select Protocol Hierarchy.

This section provides us with the distribution of protocols in the currently opened capture file, as follows:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>% Packets</th>
<th>Packets</th>
<th>% Bytes</th>
<th>Bytes/Mbls</th>
<th>End Packets</th>
<th>End Bytes</th>
<th>End Mbls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>100.00 %</td>
<td>130</td>
<td>100.00 %</td>
<td>9700</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address Resolution Protocol</td>
<td>38.08 %</td>
<td>42</td>
<td>2.29 %</td>
<td>222</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Internet Protocol Version 4</td>
<td>35.62 %</td>
<td>126</td>
<td>77.71 %</td>
<td>9478</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Transmission Control Protocol</td>
<td>23.10 %</td>
<td>124</td>
<td>10.86 %</td>
<td>9380</td>
<td>48</td>
<td>242</td>
<td>0.001</td>
</tr>
<tr>
<td>Telnet</td>
<td>1.24 %</td>
<td>2</td>
<td>2.15 %</td>
<td>209</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Using Deregistration Protocol</td>
<td>1.54 %</td>
<td>2</td>
<td>2.15 %</td>
<td>209</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Domain Name Service</td>
<td>1.54 %</td>
<td>2</td>
<td>2.15 %</td>
<td>209</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Protocol Hierarchy statistics from TelnetCapture.pcapng

The Conversations window

A conversation is a communication between two entities or endpoints. Conversations can occur over different layers, as MAC layer, network layer, and transport layer.

To view conversations, go to Statistics | Conversations.

Conversation window for WebBrowsing.pcap
If we move over to the TCP tab, we will see the options that allow us to follow TCP streams and create graphs.

**The Endpoints window**

An endpoint is just one side of the conversation and it could be Ethernet, IPv4, and other options which are visible as tabs in the **Endpoints** window. Navigate to **Statistics | Endpoints** to look at the **Endpoints** window.

When we navigate to the IPv4 tab of the **Endpoints** window, it shows us new columns such as **Country**, **City**, **Latitude**, and **Longitude**. In order to get these columns to reflect the values, we will need to configure GeoIP services first. Follow the steps mentioned later to configure GeoIP in Wireshark.

The following are the steps to configure GeoIP in Wireshark 1.12.6:

1. Download the GeoIP database. Since Wireshark does not prepackage its own set of GeoIP database(s), we will need to download a GeoIP database from http://geolite.maxmind.com/download/geoip/database/. This URL points to a freely available version of GeoIP database; however, you may also choose to buy it, if interested.

Download the Binary/gzip files for GeoLite Country and GeoLite City from the earlier-mentioned URL and extract and save these in the directory of choice. Once extracted, they will look like the following:

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoIP.dat</td>
</tr>
<tr>
<td>GeoIP.dat</td>
</tr>
<tr>
<td>GeoIPASNum.dat</td>
</tr>
<tr>
<td>GeoIPASNum.dat</td>
</tr>
<tr>
<td>GeoLiteCity.dat</td>
</tr>
<tr>
<td>GeoLiteCity.dat</td>
</tr>
</tbody>
</table>
2. Point Wireshark to the directory containing the GeoIP database. Launch Wireshark and navigate to **Edit | Preferences** and select **Name Resolution** under **User Interface** menu on the left-hand side of the window and click on **Edit** where it mentions **GeoIP database directories**, as highlighted in the following screenshot:

After clicking on **Edit**, we will be presented with the **GeoIP Database Paths** window and need to follow the steps highlighted in the following screenshot to mention the path to the directory holding the GeoIP databases, in my case D:\GeoIP.

After selecting the path, click on **OK** and then again click on **OK** in the **GeoIP Database Paths** window to apply the path changes and finally the last **OK** in the **Wireshark Preferences** window.

3. Close Wireshark and relaunch it.
4. Open any trace file of choice, navigate to the **Endpoints** window, and click on **Map**, as highlighted in the following screenshot:

By clicking on **Map**, Wireshark uses the latitude and longitude values and creates a map on the fly. The following screenshot reflects a bird's eye view, however, if we zoom in we will be able to see the yellow dots spread further to their corresponding latitude and longitude values.

Yellow dots on the map show the locations pointed by the respective latitude and longitude shown in the Endpoints window
Tweaking Wireshark

There are other interesting options under the Statistics category which we'll delve into every now and then during the course of this book.

The Expert Infos window

To open the Expert Infos window from the Menu bar navigate to Analyze | Expert Info, or simply click on the colored button on the left corner of the Status bar.

Wireshark uses Expert Infos to offer an expert advice in order to help us resolve problems and lead us to the root cause in some cases. This advice is categorized under Errors, Warnings, Notes, and Chats with Errors indicating the most severe problems and Chats showing the least.

The colored LEDs alongside these categories, as seen in the image earlier, are also present at the left corner of the Status bar indicating the level of severity for each packet.

Expert Info also has its own set of display filters as follows:

Wireshark command-line fu

In order to work conveniently with the command-line tools that come with Wireshark, it is recommended to add the path of the local Wireshark directory to the system environment variables. As we move ahead, I will assume that you've already configured the system environment variable as mentioned. Having said that, now let's look at the following more useful command-line utilities that ship with Wireshark:

- tshark
- capinfos
- editcap
- mergecap
Pass the `-h` argument with any of the command-line utilities to browse through the help options with each utility. For example, open the command prompt and run `tshark -h`.

**tshark**

The command-line version of Wireshark: `tshark` is used to capture and often display packets in typical situations when we don't have the privilege of using an interactive user interface, or when we are concerned about packet loss. Because in situations where a bulk load of traffic is flowing on the network, Wireshark's capture engine may not be able to capture at the speed with which the packets are thrown at the interface, and might crash as well. Hence, using `tshark` to capture such traffic is always a wise choice.

To look at all the options that are available with `tshark`, run the command `tshark -h`.

**Starting the capture**

If you run `tshark` without any parameters, it will start capturing on the first non-loopback interface it encounters. To look at the available interfaces, we can run the following command:

```
C:\>tshark -D
```

```
C:\>tshark -D
1. \Device\NPF\CA0A6B9947-9A5A-4B5F-87EE-908B6F7D307A (VMware Network Adapter UN nst1)
2. \Device\NPF\CA0C98ED-5F3A-49E8-9AC7-86902B3E1D9A (Ethernet)
3. \Device\NPF\CA02D275-420C-4FB4-8114-6555662E58C1 (Wi-Fi)
4. \Device\NPF\CA06F15F-0DCA-46F8-8454-589572BE95C7 (Local Area Connection)
```

Listing the interfaces with `tshark`

Simply select the interface you want to use and start capturing the traffic on that interface (in this case, 2) by running the following command:

```
C:\>tshark -i 2
```

**Stopping the capture**

To stop manually, press the combination of `Ctrl + C`.

To stop automatically, use `-a` option with a condition. The capture stops when the applied condition is met. For example, the following capture stops after 10 seconds:

```
C:\>tshark -i 2 -a duration:10
```
Saving the capture to a file

Now, there will be times when you need to save the packets captured in a file. In that case you can use the -w option:

```
C:\>tshark -i 2 -w FirstCapture.pcap
```

Using filters

You can use both display and capture filters while capturing traffic using tshark.

To use capture filters with tshark, use the -f option as given in the following:

```
C:\>tshark -i 2 -f "port bootpc" -w DHCP_Only.pcap
```

To use display filters with tshark, use the -R option as given in the following:

```
C:\>tshark -2 -R "http.request.method==GET" -r HTTP_Traffic.pcap -w HTTP_Get.pcap
```

Using the above command we're reading HTTP_Traffic.pcap, applying a display filter of http.request.method==GET and then writing the filtered packets to HTTP_Get.pcap.

Statistics

`tshark` also gives us an option to view the statistics by using the -z parameter.

To view the Protocol Hierarchy, use the following option:

```
C:\>tshark -r HTTP_Traffic.pcap -qz io,phs
```

```
C:\Users\Piyush Verma>tshark -r HTTP_traffic.pcap -qz io,phs
=====================================================================
Protocol Hierarchy Statistics
Filter:
eth                        frames:721 bytes:598880
  ip                        frames:721 bytes:598880
    tcp                      frames:86 bytes:56115
    http                     frames:10 bytes:8063
      data-text-lines       frames:6 bytes:3501
      tcp.segments           frames:10 bytes:8649
      media                  frames:9 bytes:7535
      tcp.segments           frames:22 bytes:16904
      png                     frames:21 bytes:16002
      tcp.segments           frames:1 bytes:1370
      image-gif               frames:1 bytes:733
=====================================================================
```
**capinfos**

`capinfos` is used to print the capture file's information as follows:

```plaintext
C:\Users\Plush Verma>capinfos -t HTTP_Traffic.pcap
```

Most commonly used options used with `capinfos`

The `-H` parameter is used to create hash of the capture file using the commonly used hashing algorithms [SHA1, RIPEMD160, and MD5].

We can either use these arguments individually or combine them as shown in the preceding.

You can run the `capinfos` command without passing any argument, to look at the abstract summary of the capture file, as follows:

`capinfos HTTP_Traffic.pcap`

**editcap**

This utility comes in handy when modifying capture files, such as splitting a large file into multiple file sets, removing duplicate packets from a file, or converting a capture file from one format into another.

Splitting a file into multiple file sets using `editcap`

The following example shows how to remove duplicate packets from a trace file [Duplicates.pcap]. This is generally done to save from the trouble of going over the same packets repeatedly and hence shorten the analysis time.

`C:\>editcap -d Duplicates.pcap NoDuplicates.pcap`
**mergecap**

This utility is majorly used to combine multiple capture files into a single output file. As can be seen in the following screenshot, two PCAP files were given as input to the `mergecap` utility which generated an amalgamated version named 'HTTP_Merged.pcap'.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Date/Time</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_0000_20150210215026</td>
<td>2/12/2015 11:05 AM</td>
<td>Wireshark capture file</td>
<td>328 KB</td>
</tr>
<tr>
<td>HTTP_0001_20150210215047</td>
<td>2/12/2015 11:05 AM</td>
<td>Wireshark capture file</td>
<td>282 KB</td>
</tr>
<tr>
<td>HTTP_Merged</td>
<td>2/12/2015 10:24 PM</td>
<td>Wireshark capture file</td>
<td>609 KB</td>
</tr>
</tbody>
</table>

Combining multiple HTTP capture files into HTTP_Merged.pcap

**Summary**

In this chapter, we looked at the power of using capture filters in a busy network and how to find our way through a big trace file using display filters or simply splitting it into multiple files for easy navigation. We also created new profiles in Wireshark to help us ease our day-to-day activities and learned how to use the awesome command-line utilities that are shipped with Wireshark. We will be using these as well as the advanced techniques as we move ahead further in this book. In the next chapter, we will analyze threats to LAN security.
Where to buy this book

You can buy Wireshark Network Security from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.