Analyze data network like a professional by mastering Wireshark - From 0 to 1337

Charit Mishra

What you will learn from this book

Install Wireshark and understand its GUI and all the functionalities of it
Create and use different filters
Analyze different layers of network protocols and know the amount of packets that flow through the network
Decrypt encrypted wireless traffic
Use Wireshark as a diagnostic tool and also for network security analysis to keep track of malware
Troubleshoot all the network anomalies with the help of Wireshark
Resolve latencies and bottleneck issues in the network

Who this book is written for

Are you curious to know what’s going on in a network? Do you get frustrated when you are unable to detect the cause of problems in your networks? This is where this book comes into play.

Mastering Wireshark is for developers or network enthusiasts who are interested in understanding the internal workings of networks and have prior knowledge of using Wireshark, but are not aware of all of its functionalities.


Prices do not include local sales tax or VAT where applicable
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Welcome to the World of Packet Analysis with Wireshark'
- A synopsis of the book’s content
- More information on Mastering Wireshark
Charit Mishra works as a consultant and pentester at Protiviti, one of the top global consulting firms. He enjoys his job, which involves helping clients identify security vulnerabilities, more than anything. With real hands-on experience in security, he has obtained leading industry certifications such as OSCP, CEH, CompTIA Security+, and CCNA R&S. He also holds a master's degree in computer science. He has delivered professional talks at various institutions and private organizations on information security and penetration testing. You can reach him at LinkedIn at https://ae.linkedin.com/in/charitmishra, and on Twitter at @charit0819.
Almost every device around you is connected to some other device over a network with the motive of sharing information or supporting other devices. With this small picture in your mind, what do you think is the most critical part of a network? Obviously, the channel isn't.

This book is written from a standpoint of using Wireshark to understand and troubleshoot commonly seen network anomalies. It can be the start of your journey into the world of networks/traffic/packet analysis. You can be the savior of your generation or the superhero of your team who helps people with connectivity issues, network administration, computer forensics, and so on. If your routine job requires dealing with computer networks, then this book can give you a strong head start. As the tagline says "From 0 to 1337", that is we will start from the basics gradually moving on to the advanced concepts too.

I have tried to cover the most common scenarios that you could come across while troubleshooting, along with hands-on practical cases that can make you understand the concepts better. By mastering packet analysis, you will learn how to troubleshoot all the way down to the bare wires. This will teach you to make sense of the data flowing around. You will find very interesting sections, such as troubleshooting slow networks, analyzing packets over Wi-Fi, malware analysis, and not to forget, the latest features introduced in Wireshark 2.0 in this book. Happy troubleshooting!

What this book covers

Chapter 1, Welcome to the World of Packet Analysis with Wireshark, provides you an introduction to the basics of the TCP/IP model and familiarizes you with the GUI of Wireshark along with a sample packet capture. Here, you will learn how to set up network sniffers for analysis purpose.
Preface

Chapter 2, *Filtering Our Way in Wireshark*, talks about different filtering options available in Wireshark, namely capture and display filters, and how to create and use different profiles. Make yourself comfortable with the rich interface of Wireshark and start capturing what you exactly want to.

Chapter 3, *Mastering the Advanced Features in Wireshark*, helps you look under the hood of the statistics menu in Wireshark and work with the different command-line utilities that come prepackaged with Wireshark. You will also learn how to prepare graphs, charts, packet flow diagrams, and most important of all, how to become a command-line fu master.

Chapter 4, *Inspecting Application Layer Protocols*, helps you understand and analyze the normal and unusual behavior of application-layer protocols. Here, we will briefly discuss the techniques you can use to understand the cause. We all are aware of the basics, but have you ever thought how common application-layer protocol traffic can go crazy? In this chapter, you will learn how to deal with them.

Chapter 5, *Analyzing Transport Layer Protocols*, shows how TCP and UDP protocols work, how they communicate, what problems they face, and how Wireshark can be used to analyze them. Make yourself a transport-layer doctor who can easily figure out common anomalies and prove themselves worthy.

Chapter 6, *Analyzing Traffic in Thin Air*, shows you how to analyze wireless traffic and pinpoint any problems that may follow. We will dive into the new world of wireless protocol analysis, where you can become a Wi-Fi ninja.

Chapter 7, *Network Security Analysis*, shows you how to use Wireshark to analyze network security issues, such as malware traffic, intrusion, and footprinting attempts. In this chapter, you will learn how to figure out security anomalies, catch the hackers red handed and make them cry like a baby, and experience how to solve CTF challenges.

Chapter 8, *Troubleshooting*, teaches you how to configure and use Wireshark to perform network troubleshooting. Here, you will master the art of troubleshooting network issues such as slow networks. You will also learn how to troubleshoot networking problems with the most common daily-life examples.

Chapter 9, *Introduction to Wireshark v2*, shows you the amazing features launched in the latest release of Wireshark with practical examples, such as USBpcap, intelligent scrollbar, new graphs, and much more.
Welcome to the World of Packet Analysis with Wireshark

This chapter provides you an introduction to the basics of the TCP/IP model and familiarizes you with the GUI of Wireshark along with a sample packet capture. You will be introduced to the following topics:

- What is Wireshark?
- How does it work?
- A brief overview of the TCP/IP model
- An introduction to packet analysis
- Why use Wireshark?
- Understanding the GUI of Wireshark
- The first packet capture

Introduction to Wireshark

Wireshark is one of the most advanced packet capturing software, which makes the life of system/network administrators easy and proves its usefulness among the groups of security evangelists. Wireshark is also called a protocol analyzer, which helps IT professionals in debugging network-level problems. This tool can be of great use to optimize network performance.
Wireshark runs around dissecting network-level packets and showing packet details to concerned users as per their requirement. If you are one of those who deals with packet-level networking everyday, then Wireshark is for you and can be used for multiple troubleshooting purposes.

A brief overview of the TCP/IP model

Next, it's time to discuss the most important topic in the world of networking. In order to understand how all these things stick together, we need to understand the basics of the TCP/IP model. Even the world of computers needs a set of rules and regulations to communicate, and this is taken care by the networking protocols, which govern the transmission of packets/segments/frames over a dedicated channel between hosts.

The TCP/IP model was originally known as the DoD model, and the project was regulated by United States Department of Defense. The TCP/IP model takes care of every aspect of every packet's life cycle, namely, how a packet is generated, how a single packet gets attached with a required set of information (PDU), how a packet is transmitted, how it comes to life, how it is routed through to intermediary nodes to the destination, how it is integrated back with other packets to get the whole information out, and so on.

If you have any confusion regarding the basics of networking protocols, I would recommend that you do a quick revision before proceeding ahead, as this book requires familiarity with the TCP/UDP protocols. By the time you come back, you will be able to visualize and answer all of these questions on your own.

The layers in the TCP/IP model

The TCP/IP model comprises four layers, as shown in the following diagram. Each layer uses a different set of protocols allocated to it. Every protocol has specific designated roles, and all of them are designed in such a way that they comply with industry standards.
The first layer is the **Application Layer** that directly interacts with users and other network-level protocols; it is primarily concerned with the representation of the data in an understandable format to the user. The Application layer also keeps track of user web sessions, which users are connected, and uses a set of protocols, which helps the application layer interface to the other layers in the TCP/IP model. Some popular protocols that we will cover in this book are as follows:

- **The Hyper Text Transfer Protocol (HTTP)**
- **The File Transfer Protocol (FTP)**
- **The Simple Network Management Protocol (SNMP)**
- **The Simple Mail Transfer Protocol (SMTP)**

The second layer is the **Transport Layer**. The sole purpose of this layer is to create sockets over which the two hosts can communicate (you might already know about the importance of network sockets) which is essential to create an individual connection between two devices.

There can be more than one connection between two hosts at the same instance. IP addresses and port numbers together make this possible. An IP address is required when we talk about WAN-based communication (in LAN-based communication, the actual data transfer happens over MAC addresses), and these days, a single system can communicate with more than one device over multiple channels which is possible with the help of port numbers. Apart from the restricted range of port numbers, every system is free to designate a random port for their communication.
This layer also serves as a backbone to the communication between two hosts. The most common protocols that work in this layer are TCP and UDP, which are explained as follows:

- **TCP**: This is a connection-oriented protocol, often called a reliable protocol. Here, firstly, a dedicated channel is created between two hosts and then data is transferred. Then, the sender sends equally partitioned chunks, over the dedicated channel, and then, the receiver sends the acknowledgement for every chunk received. Most commonly, the sender waits for a particular time after which it sends the same chunk again for assurance. For example, if you are downloading something, TCP is the one that takes care and makes sure that every bit is transferred successfully.

- **UDP**: This is a connection-less protocol and is often termed an unreliable form of communication. It is simple though because there is no dedicated channel created, and the sender is just concerned with sending chunks of data to the destination, whether it is received or not. This form of communication actually does not hamper the communication quality; the sole purpose of transferring the bits from a sender to receiver is fulfilled. For example, if you are playing a LAN-based game, the loss of a few bytes is not going to disrupt your gaming experience, and as a result, the user experience is not harmed.

The third layer is the **Internet Layer**, which is concerned with the back and forth movement of data. The primary protocol that works is the **IP (Internet Protocol)** protocol, and it is the most important protocol of this layer. The IP provides the routing functionality due to which a certain packet can get to its destination. Other protocols included in this layer are ICMP and IGMP.

The last layer is the **Link Layer** (often termed as the Network Interface Layer) that is close to the network hardware. There are no protocols specified in this layer by TCP/IP; however, several protocols are implemented, such as **Address Resolution Protocol (ARP)** and **Point to Point (PPP)**. This layer is concerned with how a bit of information travels inside the real wires. It establishes and terminates the connection and also converts signals from analog to digital and vice versa. Devices such as bridges and switches operate in this layer.

The combination of an IP address and a MAC address for both the client and server is the core of the communication process, where the IP address is assigned to the device by the gateway or assigned statically, and the MAC address comes from the **Network Interface Card (NIC)**, which should be present in every device that communicates with other hosts. As data progresses from the Application layer to the Link Layer, several bits of information are attached to the data bits in the form of headers or footers, which allow different layers of the TCP/IP model to coordinate with each other. The process of adding these extra bits is called data encapsulation, and in this process, a **Protocol data unit (PDU)** is created at the end of the networking model.
It consists of the information being sent along with the different protocol information that gets attached as part of the header or footer. By the time PDU reaches the bottom-most layer, it is embedded with all the required information required for the real transfer. Once it reaches the destination, the embedded header and footer PDU elements are ripped off one by one as it passes through each and every layer of the TCP/IP model as it progresses upward in the model.

The following figure depicts the process of encapsulation:

![Figure 1.1: Data encapsulation](image)

**An introduction to packet analysis with Wireshark**

Packet analysis (also known as packet sniffing or protocol analyzing) is used to intercept and capture live data as it travels over the network (Ethernet or Wi-Fi) in order to understand what is happening in the network. Packet analysis is done by protocol analyzers such as Wireshark available on the Internet. Some of these are free and some are paid for commercial use. In this book, we will use Wireshark to perform network analysis, which is an open source software and the best free-network analyzer available on the Internet.
Numerous problems can happen in today’s world of networking; for this, we need to be geared up all the time with the latest set of tools that can avail us of the ease of troubleshooting in any situation. Each of these problems will start from the packet level and can gradually grow up to a high network downtime. Even the best of protocols and services running on a system can go bad and behave maliciously. To get to the root of the problem, we need to look into the packet level to understand it better. If you need to maintain your network, then you definitely need to look into the packet level. Packet analysis can be used for the following aspects:

- To analyze network problems by looking into the packets and their specific details so that you can get a better hold over your network.
- To detect network intrusion attempts and whether there are any malicious users who are trying to get into your network, or they have already got access to something in your network.
- To detect network misuse by internal or external users by establishing firewall rules in your security appliance and then monitoring each of these rules through Wireshark.
- To isolate exploited systems so that the affected system doesn't become a pivot point for your network for malicious users.
- To monitor data in motion once it travels live in your network to have better control over the allowed and restricted categories of data. For instance, say you want to create a rule for your firewall that will block the access to Bit Torrent sites. Blocking access to them can be done from your manageable router, but knowing from where the request was originated can be easily audited through Wireshark.
- To gather and report network statistics by filtering the most specific packets as per your requirements and then creating specific capture filters for your perusal that can help you in the long run.
- Learning who is on the network and what they are doing, is there something they are not allowed to do, and is there anyone who is trying to bypass the network restrictions. All of these simple day-to-day tasks can be achieved easily through Wireshark.
- To debug client/server communications so that all the request and replies communicated between the peers on our network can be audited to maintain the integrity of your network.
- To look for applications that are sitting in the corner of your own network and eating the bandwidth. They might be making your network insecure or making it visible to the public network. Through this unnoticed application, different forms of network traffic can enter without any restrictions.
Chapter 1

- To debug network protocol implementations and any kind of anomalies present due to various misconfigurations in the current running devices.

To identify possible or malicious attacks that your network can be a victim of, to analyze them, control/supervise them, and make yourself ready for any possible malicious activity.

When performing a packet analysis, you should take care of things such as which protocols can be interpreted, which is the best software you can use according to your expertise, which protocol analyzer will best suit your network requirement. Experience does count in this field; once you start working with Wireshark, gradually you will come up with new ideas to troubleshoot and analyze your packets in a much more advanced way.

Packet sniffers can interpret common network protocols (such as IP and ICMP), transport layers (such as TCP and UDP), and application protocols (such as DNS and HTTP).

Due to the overwhelming amount of information presented by Wireshark's GUI, it might seem complex to some users and might be considered as one of its demerits. There are a few CUI/GUI tools that can solve this purpose. They are pretty simple to use and also present a simpler interface, for example, TShark, tcpdump, Fiddler, and so on.

**How to do packet analysis**

When traffic is captured, either all raw data is captured or only the header data is captured without capturing the total content of the packet. Captured information is decoded from raw data to a human-readable form, which allows users to understand the exchanged data between the networks in a much more precise manner.

**What is Wireshark?**

Wireshark is a packet-sniffing software that is used by IT professionals all around the world for analysis purpose. You can download it for free from [https://www.wireshark.org/download.html](https://www.wireshark.org/download.html).

Wireshark can be installed on a variety of platforms, including Linux, MAC, and Windows (most of the versions). This is open source software, which means that the code of the software and its required libraries can be downloaded from the same website we mentioned earlier.
One of the important key aspects of packet sniffing is where to place the packet sniffer in the physical network to achieve the maximum utilization out of it; packet sniffing is often referred to as tapping into the wire.

Tapping into the wire is not just about starting Wireshark on your system; there are a couple of things a person should know about before starting the sniffer. For instance, placing the sniffer at a proper place in the organization's infrastructure, having working knowledge of different networking devices because each of the networking devices (hubs, switches, routers, and firewalls) behave differently. It is also important to know how each of them work and how network devices handle network traffic. Placing the sniffer in the right place can impact your packet analyzing experience in a detailed manner, which in the end can lead to drastic results if done correctly.

After you have placed your sniffer, you should confirm that your NIC supports promiscuous working. By enabling this, your interface card will start learning about even those packets that are not destined or routed through your machine. A network's broadcasted traffic can be captured and analyzed by every client, which is part of the same network. Network devices broadcast multiple types of traffic that can be listened to by an interface, which supports the promiscuous mode.

The ARP protocol's traffic is broadcasted. The address resolution protocol is responsible for resolving MAC to IP addresses and vice versa. Devices such as switches send an ARP packet to all devices asking for the correct device to respond with its MAC address. Gradually, the switch will maintain a list of MAC addresses and their corresponding IP addresses, which is even termed as the CAM table (content addressable memory). Now, whenever any host wants to communicate with its other corresponding peers over the LAN, information required for the transfer is communicated to the sender from the switch. Information such as IP and MAC addresses for different devices can be easily captured and recorded through ARP traffic.

**How it works**

Wireshark comes with the libcap/Winpcap driver, which lets you switch your NIC to the promiscuous mode; the only time you don't want to sniff in the promiscuous mode is when the packets are directly, intentionally destined to your device. On a Windows-based system, you should have elevated administrator privileges to sniff and analyze the packets. There are three common step processes that every protocol analyzer follows: collect, convert, and analyze. These are described as follows:
• **Collect**: This is the first step where you choose a certain interface to listen on, and through this, you can acquire a certain amount of raw data from the network, which can be achieved by switching your interface into a promiscuous mode so that, after capturing whatever traffic is being broadcasted in your network, it can be displayed in your Wireshark GUI.

• **Convert**: This is to increase the readability of the collected binary form. Network packets can be converted by the protocol analyzer, such as Wireshark, to simple and easier formats so that people like us can have a better understanding of packets and solve our day-to-day problems easily.

• **Analyze**: In this final step, after the collection and conversion of the network packets, a step-by-step process of analyzing the data starts where we look into the specific details about the protocols and their specific configuration details. Then, we move on to host and destination addresses and the kind of information they are sharing. Rest of the analysis is left to the user's consent and how they filter and review the collected data.

If you want to get a foothold on understanding the process of packet capturing and analysis, you really need to be well versed with networking protocols and how they work because the whole communication that happens over a network is governed by various protocols, such as ARP, Dynamic Host Control Protocol (DHCP), Domain Name Service (DNS), Transmission Control Protocol (TCP), Internet Protocol (IP), HTTP, and many others.

Protocols are the rules and regulations that govern the process of communication between two network devices and control the environment under which they operate. Each of these protocols has different complexity levels depending on how and where they are being implemented. Majorly, all protocols work in the same fashion, where they send a request and wait for the confirmation, and as they receive an acknowledgement, they let the devices communicate.

After the data has been successfully transferred between them, the connections should be terminated gracefully in order to mark a communication as successful without loss of even a single bit. While the data is transferred, protocols need to maintain the integrity of the communication as well, that is, if $abc$ information is sent from the sender's side, it should be received in the same order and manner. If the bits are being tampered during the transition, this means that the protocol used isn't reliable. Analyzing all of these tasks is the basic work responsibility of any network protocol analyzer.
Capturing methodologies

Network packets can be captured through various techniques. Depending on the requirement, a protocol analyzer is placed at a certain place in network with a particular type of configuration.

Hub-based networks

Hub-based networks are the easiest ones to sniff out because you’ve the freedom to place the sniffer at any place you want, as hubs broadcast each and every packet to the entire network they are a part of. So, we don’t have to worry about the placement. However, hubs have one weakness that can drastically decrease network performance due to the collision of packets. Because hubs do not have any priority-based system for device that send packets, whoever wants to send them can just initiate the connection with the HUB (central device) and start transmitting the packets. Often, more than one devices start sending packets at the same instance. Now, as a result, the collision of the packets will happen, and the sending side will be informed to resend the previous packet. As a consequence, things such as traffic congestion and improper bandwidth utilization can be experienced.

The switched environment

Due to some restrictions present in switched-based infrastructures, packet analysis becomes a bit complex. To bypass these restrictions and make the life of administrators easy, we will talk about a couple of solutions such as port mirroring and hubbing out.

In port mirroring, once you have the command-line configuration console or web-based interface to mage you’re the access point (router/switch), then we can easily configure port mirroring.

Let's make it simpler for you with a logical illustration. For instance, let's assume that we have a 24-ports switch and 8 PCs which (PC-1 to PC-8) are connected. We are still left with more than 15 ports. Place your sniffer in any of those free ports and then configure port mirroring, which will copy all the traffic from whatever device we want to the port of our choice, where our protocol analyzer sits, which can see the whole bunch of data traveling through the mirrored port.

Once this is completely configured, we will be able to easily analyze each and every piece of information going back and forth from the mirrored port. This technique is one of the easiest among others to configure; the only thing you should know beforehand is how to configure switches with command-line interfaces. These days, admins are provided with a GUI for configuration purposes if it is the case for you to just go for it. The following figure depicts a simple demonstration of port mirroring:
Hubbing out is feasible when your switch doesn't support port mirroring. To use the technique, you have to actually plug the target PC out of the switched network, then plug your hub to the switch, and then connect your analyzer and target device to the switch so that becomes the part of the same network.

Now, the protocol analyzer and the target are part of the same broadcast domain. Your analyzer will easily capture every packet destined to target or originated from the target. But make sure that the target is aware about the data loss that can happen while you try to create hubbing out for analysis. The following figure will make it easier for us to understand the concept precisely:
ARP poisoning

This is an unethical way to capture network traffic where we try to imitate another device between two parties. Let's say, for example, we have our default gateway at 192.168.1.1 and our client is located at 192.168.1.2. Both of these devices must have maintained a local ARP cache that facilitates them to send packets without any extra overhead over the LAN. Now, the question is what kind information does the ARP cache hold, and in which form. Let me tell you, the command to view the ARP cache, which displays MAC addresses associated for a particular IP address is `arp -a`. Issuing the `arp -a` command (the same works for most of the platforms) populates a table that holds a device's IP address and its MAC address. Have a look at the following diagram which shows a normal scenario of ARP poisoning:

**Before ARP Cache**

- 192.68.1.1 – (Server)
- 192.68.1.2 – AA:BB:EE
- 192.68.1.3 – AA:BB:DD

- 192.68.1.2 – (Client)
- 192.68.1.1 – AA:BB:CC
- 192.68.1.3 – AA:BB:DD

- 192.68.1.3 – (Attacker)
- 192.68.1.1 – AA:BB:CC
- 192.68.1.2 – AA:BB:EE

Now that we've understood what is stored inside an ARP cache, let's try to poison it.

**After ARP Cache**

- 192.68.1.1 – (Server)
- 192.68.1.2 – AA:BB:DD
- 192.68.1.3 – AA:BB:DD

- 192.68.1.2 – (Client)
- 192.68.1.1 – AA:BB:DD
- 192.68.1.3 – AA:BB:DD
Now that you've understood what is the importance of the ARP protocol and how it works, we can try to poison the *arp* cache of both the default gateway and the client with the attacker's MAC address. In simple terms, we will replace the client's MAC address in the default gateway's ARP cache with the attacker's MAC address. We will do the same in the client's MAC address, replacing the default gateway's MAC address with the attacker's MAC address. As a result, every packet destined to the client from the default gateway and vice versa will be sent to the attacker's machine.
If port forwarding is already configured on the attacker's side, the received packet will be forwarded to the real intended destination, without giving any hints to the client and the default gateway that the packet is being sniffed.

![Port Mirroring Diagram]

Other than these two techniques, there is a variety of hardware available on the market, which are popularly known as taps and can be placed between any two devices to sniff and analyze the traffic. Though this technique is effective to capture network traffic in some scenarios, it should be practiced or deployed in a controlled environment because it can prove to be malicious to the internal corporate network.

**Passing through routers**

When dealing with routed environments, the main aspect of packet analyses is to place your sniffer at the right place from where we can gather the required information. Dealing with routed structures demands more skills, as sometimes you need to rethink about the placement of your sniffer. Consider a routed environment with three routers:

1. Router 1, router 2, and router 3 are working together; each of them owns 2-3 PCs.
2. Router 1 is acting like a root node while controlling its child networked nodes (router 2 and router 3).
3. Router 3 clients are not able to connect to router 1 clients.

To resolve this issue, the admin of the organization has placed the sniffer inside the router 3 area.
After a while, the admin has collected quite a good amount of packets; the admin is still not able to detect the anomaly within the network. So, he/she decides to move the sniffer to another area in the network. After placing the sniffer in the router 1 area, the admin can see quite a useful stream of packets that he/she was looking for earlier. This is quite a simple illustration of moving the sniffer around, which can be helpful in certain situations. The moral is that placing the sniffer in your networked infrastructure is quite an important task.

After reading this, I hope you would now like to see how Wireshark actually looks like, so let's take a look at the GUI of the software and how we have to initialize the process of capturing network packets.

If you do not have Wireshark installed, you can get a free copy from https://www.wireshark.org/download.html. To go through the illustrations in this book, you also need to be familiar with the interface.

**Why use Wireshark?**

I hope I am not the only one who is obsessed with the simplicity of the packet capturing scenario, which Wireshark facilitates for us. I will just quickly point out the reasons why most people prefer Wireshark to other packet sniffers:

- **User friendly**: It does count for every GUI we have ever seen or worked with, how easily the options are presented, and how convenient it is to use (I guess, even the ones who don't know about packet analysis can start capturing packets in Wireshark without any prior specialized knowledge).

- **Robustness**: The amount of information Wireshark can handle is outstanding; what I actually mean by this is software of this kind may hang or crash (because of thousands of packets that are captured and displayed every second) when trying to display the packets traveling all over the network. However, Wireshark doesn't—a big hand to Wireshark creators for how well they have structured it.

- **Platform independent**: Yeah, this one is definitely on the list. This free software can be installed on any platform that is used for computing purposes by administrators these days, whether Linux-based, Windows-based, or Macintosh-based platforms.

- **Filters**: There are two kinds of filtering options present in Wireshark:
  - You choose what to capture (capture filters)
  - You choose what to display after you've captured (display filters)
• **Cost:** Wireshark comes free, and is developed and maintained by a dedicated community. Wireshark offers some paid professional tools also. For more details refer to Wireshark's official website.

• **Support:** Wireshark is being developed very actively by a group of contributors scattered around the globe. We can sign up to the Wireshark's mailing list or we can get help from the online documentations, which can be accessed through the GUI itself; and various online forums are available to get the most effective; go to Google paid Wireshark support to know more about it.

The Wireshark GUI

Before we discuss its awesome features, let me take this opportunity to explain the history of Wireshark and how it came into existence.

Wireshark was built during the late '90s. Combs, a young college graduate from Kansas city developed Ethereal (the basic version of Wireshark), and by the time Combs developed this awesome piece of invention, he had landed himself a job where he signed a formal contract. After a few years of service, Combs decided to quit his job and to pursue his dreams by developing Ethereal further. Unfortunately, as per the legal terms, the Combs invention was part of the company's proprietary software. Despite this, Combs left the job and started working on the new version of Ethereal, which he titled Wireshark. Since 2006, Wireshark has been in active development and is being used worldwide. It supports a majority of protocols (more than 800), which are implemented in the wild today.

The installation process

Follow these steps to install Wireshark on your system:

1. In this book, I am going to you use a Mac PC; for other platforms, the installation is the same. Some OSes, such as Kali Linux, come with a preinstalled version of Wireshark.

2. So, if you are using Macintosh, then first and foremost, you need to download X11 Quartz (XQuartz-2.7.7), which will simulate an environment to run Wireshark (for Windows just download the respective executable compatible with your processor).

3. Now, you can install Wireshark (Wireshark 1.12.6 Intel 64), which we downloaded earlier in this book.

4. Once both of these are successfully installed, we need to restart our computer.
5. After the PC has been restarted, start Wireshark. As soon as the packet analyzer opens, you will see that the X11 server starts on its own. You don't need to worry about it; just leave it in the background.

6. Once it is opened completely, it will look as shown in the following screenshot:

![Figure 1.6: The Wireshark screen](image)

Before we go ahead and start the first capture, we need to get a bit familiar with the options and menus available.

There are six main parts in the Wireshark GUI, which are explained as follows:

- **Menu Bar**: This represents tools in a generalized form that are organized in the *Applications* menu.
- **Main Tool Bar**: This consists of the frequently used tools that can offer efficient utilization of the software.
- **Packet List Pane**: This window area displays all the various packets getting captured by Wireshark.
Welcome to the World of Packet Analysis with Wireshark

- **Packet Details Pane:** This window gives us details pertaining to the selected packet in the packet list pane are shown. For example, we can view source and destination IP addresses and different protocols used for communication arranged in the bottom-top approach (Link Layer to Application Layer). Information regarding the packets is listed in different categories of protocols that can be expanded to get more details for the selected packet.

- **Bytes Pane:** This shows the data in the packets in the form of hex bytes and their corresponding ASCII values; it shows the values in the form in which they travel in the wires.

- **Status Bar:** This displays details such as total packets captured.

The following screenshot will help you to identify different sections in the application, please make sure you get yourself acquainted with all of them before proceeding to further chapters.
Within the toolbar area, we have a few useful tools. I would like to give you a brief overview of some of them:

- ![interface icon]: This gives you the option to choose an interface for listening
- ![customization icon]: Through this, you can customize the capturing process
- ![start/stop/restart icon]: These are to start/stop/restart the capturing process
- ![open file icon]: This is to open a saved capture file
- ![save file icon]: This is to save the current capture in a file
- ![reload icon]: This is to reload the current capture file
- ![close icon]: This is to close the current capture file
- ![go back icon]: This is to go back to the recent most visited packet
- ![go forward icon]: This icon is to go forward to the most recently visited packet
- ![go to packet icon]: This is used to go to a specific packet number
- ![color coding toggle icon]: Toggle Color coding for the packets On/Off
- ![autoscroll toggle icon]: This is used to toggle the autoscroll on/off
- ![zoom in, out, reset zoom icon]: This is to zoom in, zoom out, and reset zoom to the default
- ![color coding change icon]: This is used to change the color coding as per requirements
- ![window narrow icon]: This is used to narrow down the window in order to capture packets
- ![display filters icon]: This is used to configure display filters to only see what is required
Even after selecting a working interface, sometimes, you won’t be able to see any packets in your packet list pane. There can be multiple reasons for this, some of which are listed as follows:

- You do not have any network traffic
- The packets traveling in the network are not destined to your device
- You do not have the promiscuous mode activated or do not have an option for the promiscuous mode

After launching the Wireshark application, you will see something like the following screenshot on our screens. Although it doesn't look so interesting at first glance, what makes it interesting are the packets that are flowing around. Yeah, I am talking about capturing packets.

![Figure 1.7: The Wireshark capture screen](image)

**Starting our first capture**

As you’ve been introduced to the basics of Wireshark and since you have learned how to install Wireshark, I feel you are ready to initiate your first capture. I will be guiding you through the following series of steps to start/stop/save you first Wireshark capture:

1. Open the Wireshark application.
2. Choose an interface to listen to.
3. Before you click on **Start**, we have the **Options** button, which gives us the advantage of customizing the capture process; but as of now, we will be using the default configuration.

Make sure that the **Promiscuous** mode is activated so that we can capture the traffic that is not destined to our machine.
4. Click on the **Start** button to initiate the capturing process.
5. Open your browser.
6. Visit any website you want to.

![The Wireshark website](image)

Figure 1.10: The Wireshark website

7. Switch back to the Wireshark screen; if everything goes well, you should be able to see a numerous packets getting captured in your Wireshark GUI inside the packet list pane.

To stop the capture, you can just click on the **stop capture** button in the toolbar area or you can click on **Stop** under the **Capture** menu bar.
8. I know there is an overwhelming amount of information you will see by now, but don’t worry about it. I am here to make it simple for you.

9. The real process of packet analysis starts when you have captured packets—I mean packet filtering. We will be discussing packet filtering in detail in the upcoming chapters.

10. Now, the last step is to save the capture file for later use:

   ![Capture File Save](image)

   11. Save your file with the default .pcapng extension in your folder.
If you have read all the steps all the way up to this point, I would encourage you to create your first capture file.

**Summary**

This chapter lays the foundation of basic networking concepts along with an introduction of the Wireshark GUI. Wireshark is a protocol analyzer that is used worldwide by IT professionals to capture and analyze network-level packets.

The TCP/IP model has four layers: the Application Layer, Transport Layer, Network Layer, and Link Layer. Data gets encapsulated as it passes on from one layer to another; the resulting packet at the bottom is called a complete PDU, which actually travels over the channel.

To install Wireshark, you just need to visit [http://www.wireshark.org](http://www.wireshark.org) and then download the appropriate version of this open source software. The Wireshark community is governed by real-world geeks; this can be a good source of learning and for troubleshooting purposes.

The Wireshark GUI is user friendly, robust, and platform independent; even new IT professionals can easily adapt the tool.

One important aspect of protocol analyzing is to place the sniffer at the right place; every organization’s infrastructure is different from another, where we might need to apply different techniques in order to get the right packets to use.

Hubbing out, port mirroring, ARP poisoning, and tapping are some of those useful techniques that can be used to monitor and analyze traffic in different situations.

There are six main parts in the Wireshark tool window: **Menu Bar, Main Tool Bar, Packet List Pane, Packet Details Pane, Bytes Pane,** and **Status Bar**.

Using the back/forward key during a packet analysis scenario can be really useful. One should know about all the tools that are displayed in the main toolbar area.

In the next chapter, you will learn how to work with different kinds of filters available in Wireshark.
Practice questions
Q.1 How many layers are there in the TCP/IP? Name them.
Q.2 Which layer in the TCP/IP model handles Layer 2 addresses?
Q.3 The Link Layer is also called?
Q.4 The HTTP protocol uses TCP or UDP?
Q.5 IP, ICMP, and ________ are the protocols in the Internet Layer
Q.6 How many parts of the Wireshark window do you know?
Q.7 ARP is a Layer 3 protocol—true/false?
Q.8 Does the TCP protocol follow a three-way handshake?
Q.9 The Port Mirroring technique is possible through switches only—True/False?
Q.10 The Hubbing out technique uses a router to isolate a PC from it peers—true/false?
Q.11 TCP is an unreliable protocol—true/false?
Q.12 Install Wireshark and start a sample capture using your wireless interface. Save your capture file on the desktop with the name first.pcap, and close Wireshark.
Q.13 Open your first.pcap capture file in Wireshark and check how many packets you captured in total.
Q.14 Which pane displays information in the HEX and ASCII form for each packet we've captured?
Q.15 Switch off the promiscuous mode from the capture options window and observe whether you are still able to receive packets from other devices or not.
Where to buy this book

You can buy Mastering Wireshark 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.