It can be daunting trying to learn how to use a new piece of software, even if you already possess a fair amount of knowledge regarding information technology. And there are few other programs that have such a steep learning curve as Wireshark. In my humble opinion, it's one of the best packet sniffers and protocol analyzers available, and it's truly mind blowing how much detailed information it can gather. Believe it or not, a competent Wireshark user can even see personal information that is transmitted in plain text, provided they are using a man-in-the-middle attack or redirecting other users' traffic via DNS-based attacks. It's uses are seemingly endless, but before we dig into the inner workings of this impressive piece of software, we need to lay some groundwork.

What is Wireshark?

Wireshark is a program that has the ability to record and analyze every last bit flowing through a network interface. In the I.T. world, it is sometimes called a protocol analyzer, packet sniffer, packet capture utility, or packet analyzer – and these terms are synonymous and are mostly interchangeable for all but the most anal network engineers.

The tool is actually rather sophisticated and allows network administrators and hackers to capture specific types of traffic. For example, if a network administrator wanted to see which websites and individual computer or group of computers were accessing, then he/she might run a packet capture for only HTTP and DNS traffic. On the other hand, a Wireshark user could choose to capture *every single packet* flowing over an interface, and then sift through all the data at their own leisure.

Not only is the software very powerful, but it is open source software that can be used free of charge. If you're new to networking protocols already, the amount of data that it collects in real time might seem very intimidating. Just remember not to abuse the software, because wiretapping is illegal and it would raise some strange questions if you got caught using it a work, school, or on any network you don't personally own. But we're going to delve a little deeper to show you how to run some captures on your home network.

What is Wireshark Used for?

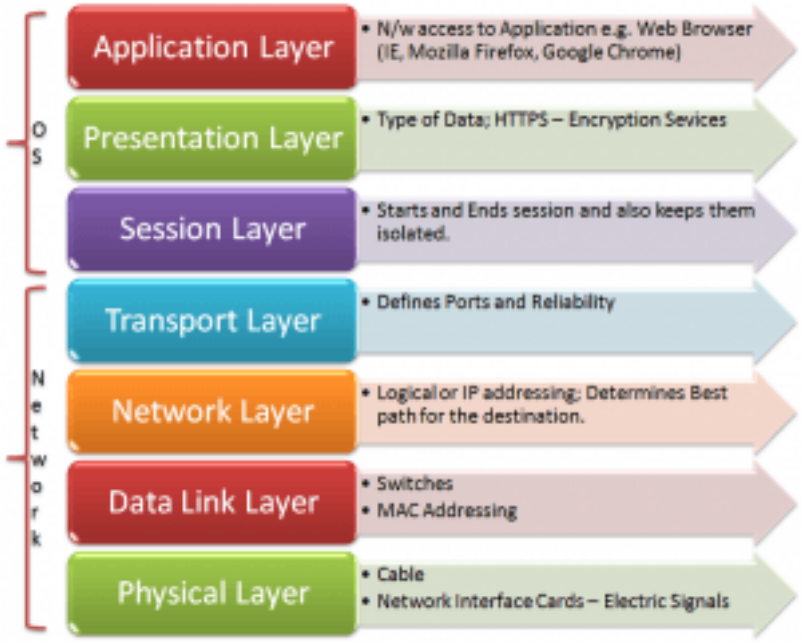
As mentioned previously, the uses for Wireshark are virtually limitless, and are only bound by the number of protocols and individual computing systems that a hacker or administrator wishes to analyze. But understand that protocol analyzers are a doubled edged sword. Like an inanimate weapon, the choice whether it's used for good or evil depends on who's wielding it. Originally, it was created as a tool to allow network engineers to troubleshoot network problems.

In addition, it was also used as a security tool. For example, an administrator may wish to verify that security software, such as a VPN tunnel, is truly encrypting data. By capturing

the data and sifting through it in Wireshark, an administrator will be able to see if the payload was sent in plain text. The information it collects can also be used to debug network protocols that are invisible to the average user, and that makes it one of the best tools to learn different TCP/IP algorithms and to actually see how these protocols work at a highly technical level. But as you might expect, having all of this information at your fingertips makes it easy to snoop through users' data, which is one reason why hackers like it so much.

Wireshark will help them uncover information about hosts that isn't always visible with other reconnaissance methods. Consider that ping sweeps, which are common and simple ways to view other active network hosts on a LAN, are fundamentally flawed. A computer system may be configured to drop pings (ICMP packets) or to never respond to ICMP echo requests. This means that a ping sweep may not see *all* of the hosts and interfaces on a given subnet. However, these systems' IP addresses and traffic will be visible if an attacker can capture them with Wireshark – especially if they can capture data from a single point of entry/exit over the LAN (like the LAN's default gateway interface).

Wireshark's Features



Wireshark is absolutely loaded with features, and there are so many of them that we couldn't hope to list them all without writing a thick, comprehensive manual. However, at the very least, we can get a high level understanding of its features as follow:

- The ability to perform deep inspection on *hundreds* of network protocols
- A three pane packet browsing interfaces

- The ability to drill down through the different layers of a packet and display information from the header, various wrappers, and the payload
- Multi-platform support that includes Windows, Linux, Solaris, FreeBSD, NetBSD, OS X, among others (e.g. all the major operating system platforms)
- Network data that has been captured can be viewed in a convenient GUI or saved in databases that can be queried through the command line
- *Extremely* powerful, flexible, and robust filtering mechanisms that capture or display only select network protocol traffic
- VoIP features that are able to analyze voice data that will reveal information such as the start time of a call, the stop time of a call, who initiated the call, various protocols information (H323, ISUP, UNISTIM, SIP, MGCP, etc.), the state of the call (call setup info, ringing, in call, canceled, completed, rejected, etc.), and IP addresses
- The ability to actually replay a captured VoIP call for select codecs
- Coding that interfaces well with other file formats such as libpcap, tcpdump, Pcap NG, and many, many others
- Capture and decompress gzip files on the fly
- Read live data from a smattering of types of physical interfaces including Ethernet, Wireless (802.11x), PPP, HDLC, ATM, USB, Bluetooth, Fram Relay, Token Ring, FDDI (Fiber Distributed Data Interface), and others – really just about anything you can imagine
- Some special case decryption support for several popular protocols including WEP, WPA, WPA2, SSL/TLS, SNMPv3, ISAKMP, and Ipsec
- The GUI includes a feature to set different types of traffic to different color schemes to more easily sift through and analyze data
- All of the collected information can be exported to PostScript, CSV files, plain text, or XML

Remember that this certainly isn't a comprehensive list of its features, but it should give you a good idea just how powerful this tool is and some of its major capabilities. Also understand that in order to use some of the more advanced features (like capturing a VoIP call and listening to the audio contents) is no easy task for a hacking novitiate.

Though Wireshark allows you to *see* this data, you'll also want to have an understanding of the various networking protocols and how they operate. There's not really a shortcut or easy way around this, and it just takes some good 'ol fashioned studying. That said, for some of the more simple tasks, all you need is a simple step by step guide (as we'll provide you with shortly).

Methods of Capturing User Data

Before we begin our capture, you need to understand that each capture runs on a specified *interface*. For example, if you wanted to capture all of the traffic through your Ethernet connection, you would need to run a capture on that specific interface. Likewise,

to capture wireless packets, you would need to run a capture on an appropriate wireless interface.

At this point you might be wondering how attackers use Wireshark to initiate attacks and gather information. After all, what's the point of running a capture on your system's local Ethernet interface? In truth, it won't see any data transmitted by other computers to remote systems such as a public web server. To see such data that is transmitted by other hosts or the target of an attack, the attacker would need to trick the remote system into sending their data to the attacker's computer, and then forward that data to the intended destination.

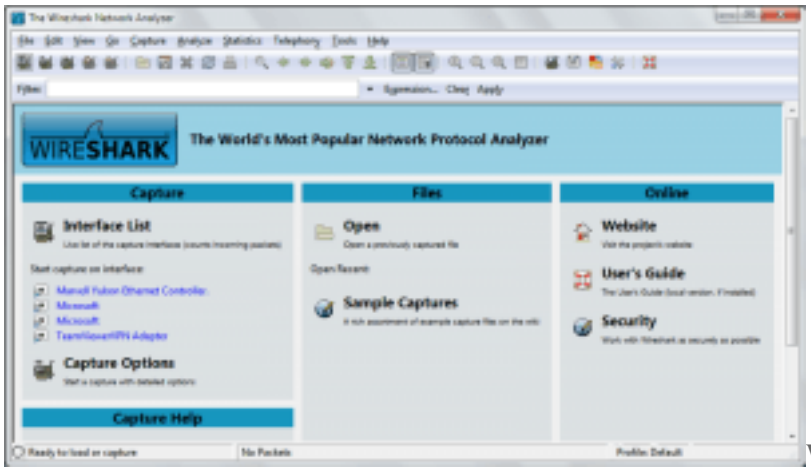
This technique is most commonly facilitated with a man-in-the-middle attack, whereby the hacker tricks a target system into thinking that their MAC address (a layer 2 address), is the default gateway's address. This is called MAC spoofing, but there are other ways to accomplish the same thing. For example, if an attacker had access to a local subnet's DHCP server (Dynamic Host Configuration Protocol), they could employ some nasty tricks that leverage DNS.

You see, in a home environment, the default gateway – which is most typically a SOHO wireless router – supplies the DHCP parameters such as the IP addresses, subnet mask, and even the default DNS servers. An attacker could change the DNS parameter to a server that he/she controls. Then, the attacker has supreme control over where a users' data goes on the Internet. For example, they could insert a bogus DNS record that causes www.google.com to resolve to an IP address of a server or workstation that the attacker also controls.

Then, every time a user sends data to Google, it first travels through the attackers network interface – where they are running a packet capture. Now consider how much data an attacker could gather if they broke into an ISP's network. ISPs serve hundreds of thousands – even millions – of users every day. An attacker that had illegal access to an interface on an ISP network could run a capture with criteria that tracks all data destined to a certain website, such as www.Facebook.com, in an attempt to harvest login credentials. Though Facebook uses HTTPS to encrypt data, realize that this is only an example.

Today we're not going to run a capture that targets a remote system – doing so would be unethical, illegal, and frankly a little too difficult if you've never used Wireshark before. So, before we run our first captures, let's start with the user interface.

Understanding the Wireshark Interface



When you fire up Wireshark for the first time, things can seem to be a little confusing. Before we dig into the the juicy details of how to run a packet capture, let's first take a moment to familiarize ourselves with the interface. First off, note that I am referring to the latest version, which is Wireshark 2.0.3 (which was made available in 2016).

After you have downloaded, installed, and launched the software, you will be presented with the main window. The main menu, which has sections including, file, edit, view, go, capture, analyze, and so forth is the best interface for newbies to use. We'll just take a moment to gain a high level understanding of this menu so you'll know where all of the commands are stored in the interface.

- **File:** allows a user to open, save, print, export, and even merge capture files
- **Edit:** allows a user to set their global preferences, find specific packets, mark or reference packets based on timestamps, and make configurations
- **View:** allows a user to sift through data, indicate color preferences for different types of traffic, and show/collapse packet details through a tree structure
- **Capture:** this feature is arguably the most important for novices, because it is here that all captures are started and stopped. In addition, users have the power to edit existing captures
- **Analyze:** helps users sort data with display filters, enable/disable protocol dissection, and manipulate certain types of traffic such as TCP streams
- **Statistics:** the statistics menu includes features that run statistical computations to generate information such as the total number of packets captured, protocol hierarchy data, and more
- **Telephony:** though difficult to use for new users, the telephony menu includes options to help manipulate captured voice traffic
- **Wireless:** naturally, this menu contains tools to help users capture and manipulate 802.11 protocol information

You'll also notice that there are a myriad of buttons underneath this menu. For now, just be aware that this menu is the best place to start. In my opinion, the most important menu items that new users should be focused on are **file, edit, view, and capture** – because this

will allow a novice to run their first capture, edit some of its settings, save the capture, and view its contents.

Running Your First Capture

Today we're going to keep things simple. We're just going to run some simple captures on ***your own local network that you own and control***. For the purposes of this demonstration, you will need the following:

- A computer with an Ethernet connection
- A computer in which you have downloaded and installed the GUI version of Wireshark
- Network connectivity (e.g. an IP address, default gateway, Internet connection, etc.)

After you have launched Wireshark, you should notice that there is a list of interfaces under the **Capture** column. Alternatively, you can use the **Capture** button in the menu as we discussed previously. Let's assume that you just clicked on the interfaces name in the **Capture** column. Doing so would immediately start to record and capture all of the data leaving and entering your Ethernet interface.

Even if you don't have a web browser running, you'll notice that the screen become populated with a myriad of information and different colored rows and columns. The data you are seeing is network protocol data, and it's likely that you never knew all of this was going on in the background. Also note that your computer will continue to monitor and record *all* of the data flowing through your interface until you stop the capture.

Stopping Your First Capture

You won't want to let your capture run indefinitely, and by now you likely have more protocol data than you know what to do with. To start analyzing and sifting through the data you have collected, you'll need to shut down your capture. Look for the fourth button from the left, located just below the **View** menu, and you'll see an icon with a white 'X' surrounded by a red circle. Alternatively, you can shut down your capture from the **Capture** menu.

Take a few moments to look through the data you've collected. If you want, you can begin a new capture and then browse to a website so you can view the details of the HTTP request between your computer and a webserver. Also, you'll likely see some familiar protocols like DNS, ARP, and DHCP among others.

Advanced Capture Options

Next, we're going to take a brief look at how you can fine-tune the parameters of your capture to weed out types of data that you don't necessarily want to analyze. All you need to do is browse through the **Capture** menu and then select **Options**. Then, a dialog box will open that allows you to change various details. Clicking on the **Capture Filter** option will allow you to only capture certain types of traffic, such as TCP traffic, UDP traffic, or traffic streams that operate on a specific port. For example, to capture only DNS traffic, you would want to filter by port 53. Then, simply rerun the capture again.

Wireshark, Certifications, and Practical Applications

Wireshark is an undoubtedly useful tool with as many uses as you can imagine. And one of it's uses is helping you pass various certification exams, such as the CEH, Network+, CCNA, and Security+ exams. In fact, Wireshark (and packet sniffers/protocol analyzers in general) are part of many security exams content and questions.

For instance, if you hope to pass the Security+ exam, you're going to need to know what a protocl analyzer is and how it works. And as they say, experience is the best teacher. I would highly recommend downloading a free version of Wireshark and playing around with it to get an idea of what information it can capture from a networking interface.

But Wireshark isn't the only method of capturing packets. As a penetration tester, you're going to need to know how to use several different types of packet capture tools. For example, Cisco ASA's have a command line utility that allows you to capture traffic on any of the ASA's networking interfaces. I can't tell you how many times I ran a packet capture on an ASA to verifiy that certain traffic was or wasn't passing through the firewall as a means to test and verify the firewall rules were configured appropriately.

In addition, Wireshark and other packet capture tools can help you troubleshoot different types of network protocols. As an example, note that I've spent time in the past using Wireshark to check a network interface to validate VPN tunnel negotiation and to check various handshaking processes.

These skills and knowledge are invaluable to a penetration tester, and Wireshark can help you understand how various protocols work on a highly detailed and technical level. If you're studying for the CCNA exam, being able to actually *see* protocol packets, such as ARP requests, will help you understand how protocols and computer systems 'talk' to each other. Being able to see these processes is critical to your success, because otherwise it all seems like theoretical knowledge that just happens 'invisibly' in the background.

Furthermore, note that if you get hired for any kind of I.T. security role and you show up on day one without knowing what Wireshark is, you're likely going to become the victim of a lot of jokes. Your coworkers will laugh you out of the office, and you'll look

extremely inexperienced. The bottom line is that Wireshark (and similar tools) are used on an extremely frequent basis in the security realm, so you'd better know what your doing before you get hired. I'd highly recommend using Wireshark to pick apart different types of traffic before sitting your exams, too.

Final Thoughts

Wireshark is an incredibly powerful tool used to capture and analyze just about every type of protocol traffic you can think of. It can even be used to verify if your encryption technologies are actually doing their job, or if a configuration error is sending data in plain text. The bottom line is that knowing how to use Wireshark is a critical skill crucial to your success in any security discipline as well as for standard network engineering. Last but not least, it's a lot of fun to use. You wouldn't believe all the data that gets passed around between computers in the background, and Wireshark will help lift the veil so you can watch traffic move across an interface in real time.