**Network traffic Analysis with Tcpdump, Analysis with Wireshark**

**Intro:** Tcpdump is a command-line packet sniffer that can directly capture and interpret data frames from a file or network interface. It was built for use on any Unix-like operating system and had a Windows twin called WinDump. It is a potent and straightforward tool used on most Unix-based systems. It does not require a GUI and can be used through any terminal or remote connection, such as SSH.

TCPDump is available for most Unix systems and Unix derivatives, such as AIX, BSD, Linux, Solaris, and is supplied by many manufacturers already in the system. Due to the direct access to the hardware, we need the root or the administrator's privileges to run this tool. For us that means we will have to utilize sudo to execute TCPDump as seen in the examples below. TCPDump often comes preinstalled on the majority of Linux operating systems.

Validating if the package exists.

* Which tcpdump, command use to validate on hour. Often it can be found in /usr/sbin/tcpdump.
* A black background with green and white text

  Description automatically generated
* However, if the package does not exist, we can install it with:
* Sudo apt install tcpdump

We can run the tcpdump package with the --version switch to check our install and current package version to validate our install.

* Sudo tcpdump –version

A screenshot of a computer

Description automatically generated

**Traffic Captures with Tcpdump**

Because of the many different functions and filters, we should first familiarize ourselves with the tool's essential features. Let us discuss some basic TCPDump options, demo some commands, and show how to save traffic to PCAP files and read from these.

**Basic Capture Options**

Below is a table of basic Tcpdump switches we can use to modify how our captures run. These switches can be chained together to craft how the tool output is shown to us in STDOUT and what is saved to the capture file. This is not an exhaustive list, and there are many more we can use, but these are the most common and valuable.

| **Switch Command** | **Result** |
| --- | --- |
| D | Will display any interfaces available to capture from. |
| i | Selects an interface to capture from. ex. -i eth0 |
| n | Do not resolve hostnames. |
| nn | Do not resolve hostnames or well-known ports. |
| e | Will grab the ethernet header along with upper-layer data. |
| X | Show Contents of packets in hex and ASCII. |
| XX | Same as X, but will also specify ethernet headers. (like using Xe) |
| v, vv, vvv | Increase the verbosity of output shown and saved. |
| c | Grab a specific number of packets, then quit the program. |
| s | Defines how much of a packet to grab. |
| S | change relative sequence numbers in the capture display to absolute sequence numbers. (13248765839 instead of 101) |
| q | Print less protocol information. |
| r file.pcap | Read from a file. |
| w file.pcap | Write into a file |

**Man Page Utilization**

To see the complete list of switches, we can utilize the man pages:



A computer screen shot of white text

Description automatically generated

Here are some examples of basic Tcpdump switch usage along with descriptions of what is happening:

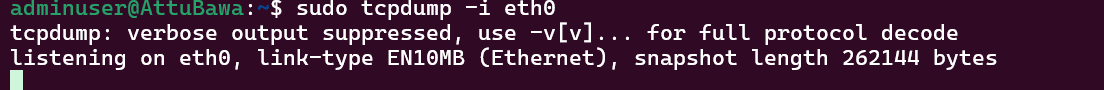
Listing Available interfaces:

A screenshot of a computer program

Description automatically generated

The above command sudo tchpdump -D calls tcpdump using sudo privileges and lists the usable network interfaces. We can choose one of these network interfaces and tell tcpdump which interfaces it should listen to.

**Choosing an Interface to Capture From**

****

When utilizing the -e switch, we are tasking tcpdump to include the ethernet headers in the capture's output along with its regular content. We can see this worked by examining the output. Usually, the first and second fields consist of the Timestamp and then the IP header's beginning. Now it consists of Timestamp and the source MAC Address of the host.

**Display the Ethernet Header**

A computer screen with white text

Description automatically generated

By issuing the -X switch, we can see the packet a bit clearer now. We get an ASCII output on the right to interpret anything in clear text that corresponds to the hexadecimal output on the left.

**Include ASCII and Hex Output**

A screenshot of a computer screen

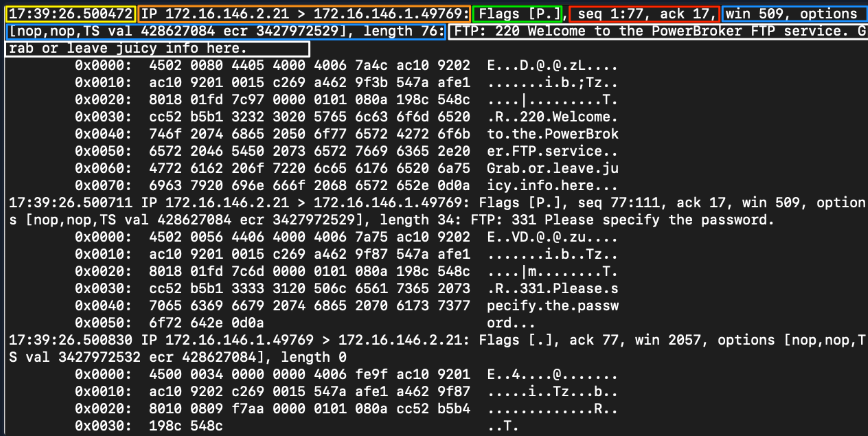
Description automatically generated

Pay attention to the level of detail in the output above. We will notice that we have information on the IP header options like time to live, offset, and other flags and more details into the upper layer protocols. Below, we are combining the switches to craft the output to our liking.

**Tcpdump Output**

When looking at the output from TCPDump, it can be a bit overwhelming. Running through these basic switches has already shown us several different views. We are going to take a minute to dissect that output and explain what we are seeing. The image and table below will define each field. Keep in mind that the more verbose we are with our filters, the more detail from each header is shown.

**Tcpdump Shell Breakdown**



| **Filter** | **Result** |
| --- | --- |
| Timestamp | Yellow The timestamp field comes first and is configurable to show the time and date in a format we can ingest easily. |
| Protocol | Orange This section will tell us what the upper-layer header is. In our example, it shows IP. |
| Source & Destination IP.Port | Orange This will show us the source and destination of the packet along with the port number used to connect. Format == IP.port == 172.16.146.2.21 |
| Flags | Green This portion shows any flags utilized. |
| Sequence and Acknowledgement Numbers | Red This section shows the sequence and acknowledgment numbers used to track the TCP segment. Our example is utilizing low numbers to assume that relative sequence and ack numbers are being displayed. |
| Protocol Options | Blue Here, we will see any negotiated TCP values established between the client and server, such as window size, selective acknowledgments, window scale factors, and more. |
| Notes / Next Header | White Misc notes the dissector found will be present here. As the traffic we are looking at is encapsulated, we may see more header information for different protocols. In our example, we can see the TCPDump dissector recognizes FTP traffic within the encapsulation to display it for us. |

**Analysis with Wireshark**

Wireshark is a free and open-source network traffic analyzer much like tcpdump but with a graphical interface. Wireshark is multi-platform and capable of capturing live data off many different interface types (to include WiFi, USB, and Bluetooth) and saving the traffic to several different formats. Wireshark allows the user to dive much deeper into the inspection of network packets than other tools. What makes Wireshark truly powerful is the analysis capability it provides, giving a detailed insight into the traffic.

Depending on the host we are using, we may not always have a GUI to utilize traditional Wireshark. Lucky for us, several variants allow us to use it from the command line.

**Features and Capabilities:**

* Deep packet inspection for hundreds of different protocols
* Graphical and TTY interfaces
* Capable of running on most Operating systems
* Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, among others
* Decryption capabilities for IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
* Many many more...

**Requirements for Use**

Wireshark requires the following for use:

**Windows:**

* The Universal C Runtime. This is included with Windows 10 and Windows Server 2019 and is installed automatically on earlier versions if Microsoft Windows Update is enabled. Otherwise, KB2999226 or KB3118401 must be installed.
* Any modern 64-bit AMD64/x86-64 or 32-bit x86 processor.
* 500 MB available RAM. Larger capture files require more RAM.
* 500 MB available disk space. Capture files require additional disk space.
* Any modern display. 1280 × 1024 or higher resolution is recommended. Wireshark will make use of HiDPI or Retina resolutions if available. Power users will find multiple monitors useful.
* A supported network card for capturing:
  + Ethernet. Any card supported by Windows should work.
  + 802.11. See the Wireshark wiki page. Capturing raw 802.11 information may be difficult without special equipment.
* To install, download the executable from wireshark.org, validate the hash, and install.

**Linux:**

* Wireshark runs on most UNIX and UNIX-like platforms, including Linux and most BSD variants. The system requirements should be comparable to the specifications listed above for Windows.
* Binary packages are available for most Unix and Linux distributions.
* To validate if the package exists on a host, use the following command:
* **TShark VS. Wireshark (Terminal vs. GUI)**
* Both options have their merits. TShark is a purpose-built terminal tool based on Wireshark. TShark shares many of the same features that are included in Wireshark and even shares syntax and options. TShark is perfect for use on machines with little or no desktop environment and can easily pass the capture information it receives to another tool via the command line. Wireshark is the feature-rich GUI option for traffic capture and analysis. If you wish to have the full-featured experience and work from a machine with a desktop environment, the Wireshark GUI is the way to go.
* **Basic TShark Switches**

| **Switch Command** | **Result** |
| --- | --- |
| D | Will display any interfaces available to capture from and then exit out. |
| L | Will list the Link-layer mediums you can capture from and then exit out. (ethernet as an example) |
| i | choose an interface to capture from. (-i eth0) |
| f | packet filter in libpcap syntax. Used during capture. |
| c | Grab a specific number of packets, then quit the program. Defines a stop condition. |
| a | Defines an autostop condition. Can be after a duration, specific file size, or after a certain number of packets. |
| r (pcap-file) | Read from a file. |
| W (pcap-file) | Write into a file using the pcapng format. |
| P | Will print the packet summary while writing into a file (-W) |
| x | will add Hex and ASCII output into the capture. |
| h | See the help menu |

**Termshark**

Termshark is a Text-based User Interface (TUI) application that provides the user with a Wireshark-like interface right in your terminal window.

A screenshot of a computer

Description automatically generated

Termshark can be found at [Termshark](https://github.com/gcla/termshark). It can be built from the source by cloning the repo, or pull down one of the current stable releases from https://github.com/gcla/termshark/releases , extract the file, and hit the ground running.

For help navigating this TUI, see the image below.

**Termshark Help**

A screenshot of a computer

Description automatically generated

To start Termshark, issue the same strings, much like TShark or tcpdump. We can specify an interface to capture on, filters, and other settings from the terminal. The Termshark window will not open until it senses traffic in its capture filter. So give it a second if nothing happens.

**Wireshark GUI Walkthrough**

Now that we have spent time learning the art of packet capture from the command line let's spend some time in Wireshark. We will take a few minutes to examine what we are looking at in the output below. Let's dissect this view of the Wireshark GUI.

**Wireshark GUI**

A screenshot of a computer

Description automatically generated

Three Main Panes: See Figure above

1. Packet List: Orange
   * In this window, we see a summary line of each packet that includes the fields listed below by default. We can add or remove columns to change what information is presented.
     + Number- Order the packet arrived in Wireshark
     + Time- Unix time format
     + Source- Source IP
     + Destination- Destination IP
     + Protocol- The protocol used (TCP, UDP, DNS, ETC.)
     + Information- Information about the packet. This field can vary based on the type of protocol used within. It will show, for example, what type of query It is for a DNS packet.
2. Packet Details: Blue
   * The Packet Details window allows us to drill down into the packet to inspect the protocols with greater detail. It will break it down into chunks that we would expect following the typical OSI Model reference. The packet is dissected into different encapsulation layers for inspection.
   * Keep in mind, Wireshark will show this encapsulation in reverse order with lower layer encapsulation at the top of the window and higher levels at the bottom.
3. Packet Bytes: Green
   * The Packet Bytes window allows us to look at the packet contents in ASCII or hex output. As we select a field from the windows above, it will be highlighted in the Packet Bytes window and show us where that bit or byte falls within the overall packet.
   * This is a great way to validate that what we see in the Details pane is accurate and the interpretation Wireshark made matches the packet output.
   * Each line in the output contains the data offset, sixteen hexadecimal bytes, and sixteen ASCII bytes. Non-printable bytes are replaced with a period in the ASCII format.

**Other Notable Features**

When looking at the Wireshark interface, we will notice a few different option areas and radial buttons. These areas are control points in which we can modify the interface and our view of the packets in the current capture. See Figure below

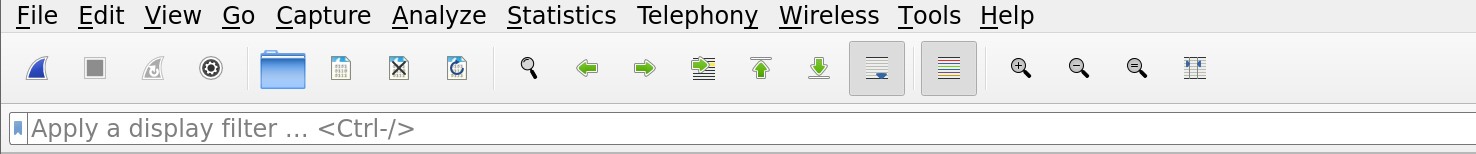
**Wireshark Menu**

A screenshot of a computer

Description automatically generated

**The Basics**

**The Toolbar**



Wireshark's Toolbar is a central point to manage the many features Wireshark includes. From here, we can start and stop captures, change interfaces, open and save .pcap files and apply different filters or analysis add-ins.

**How to Save a Capture**

Let's say we need to capture what we have in our window currently for troubleshooting later. Saving a capture is super simple:

* Select File ⇢ save OR
* From the toolbar, select the file option and choose where to save the file and in what format.
* **Pre-capture and Post-capture Processing and Filtering**
* While capturing traffic with Wireshark, we have several options regarding how and when we filter out traffic. This is accomplished utilizing Capture and Display filters. The Former initiated before the capture starts and the latter during or after capture is complete. While Wireshark has a bunch of useful baked-in functionality, it is worth mentioning that it has a bit of trouble handling large captures. The more packets captured, the longer it will take Wireshark to run the display or analysis filter against it. It can take from just a couple of seconds to a few minutes if it completes at all. If we are working with a large pcap file, it may be best to break it up into smaller chunks first.

Here is a table of common and helpful capture filters with a description of each:

| **Capture Filters** | **Result** |
| --- | --- |
| host x.x.x.x | Capture only traffic pertaining to a certain host |
| net x.x.x.x/24 | Capture traffic to or from a specific network (using slash notation to specify the mask) |
| src/dst net x.x.x.x/24 | Using src or dst net will only capture traffic sourcing from the specified network or destined to the target network |
| port # | will filter out all traffic except the port you specify |
| not port # | will capture everything except the port specified |
| port # and # | AND will concatenate your specified ports |
| portrange x-x | portrange will grab traffic from all ports within the range only |
| ip / ether / tcp | These filters will only grab traffic from specified protocol headers. |
| broadcast / multicast / unicast | Grabs a specific type of traffic. one to one, one to many, or one to all. |

**Applying a Capture Filter**

Before we apply a capture filter, let us take a look at the built-in filters. To do so: Click on the capture radial at the top of the Wireshark window → then select capture filters from the drop-down.

**Filter List**

A screenshot of a computer

Description automatically generated

From here, we can modify the existing filters or add our own.

To apply the filter to a capture, we will: Click on the capture radial at the top of the Wireshark window → then select Options from the drop-down → in the new window select the drop-down for Capture filter for selected interfaces or type in the filter we wish to use. below the red arrow in the picture below

**Applying A Capture Filter**

A screenshot of a computer

Description automatically generated

Here is a table of common and helpful display filters with a description of each:

| **Display Filters** | **Result** |
| --- | --- |
| ip.addr == x.x.x.x | Capture only traffic pertaining to a certain host. This is an OR statement. |
| ip.addr == x.x.x.x/24 | Capture traffic pertaining to a specific network. This is an OR statement. |
| ip.src/dst == x.x.x.x | Capture traffic to or from a specific host |
| dns / tcp / ftp / arp / ip | filter traffic by a specific protocol. There are many more options. |
| tcp.port == x | filter by a specific tcp port. |
| tcp.port / udp.port != x | will capture everything except the port specified |
| and / or / not | AND will concatenate, OR will find either of two options, NOT will exclude your input option. |

**Applying a Display Filter**

Applying a display filter is even easier than a capture filter. From the main Wireshark capture window, all we need to do is: select the bookmark in the Toolbar → , then select an option from the drop-down. Alternatively, place the cursor in the text radial → and type in the filter we wish to use. If the field turns green, the filter is correct

A screenshot of a computer

Description automatically generated

**Plugins**

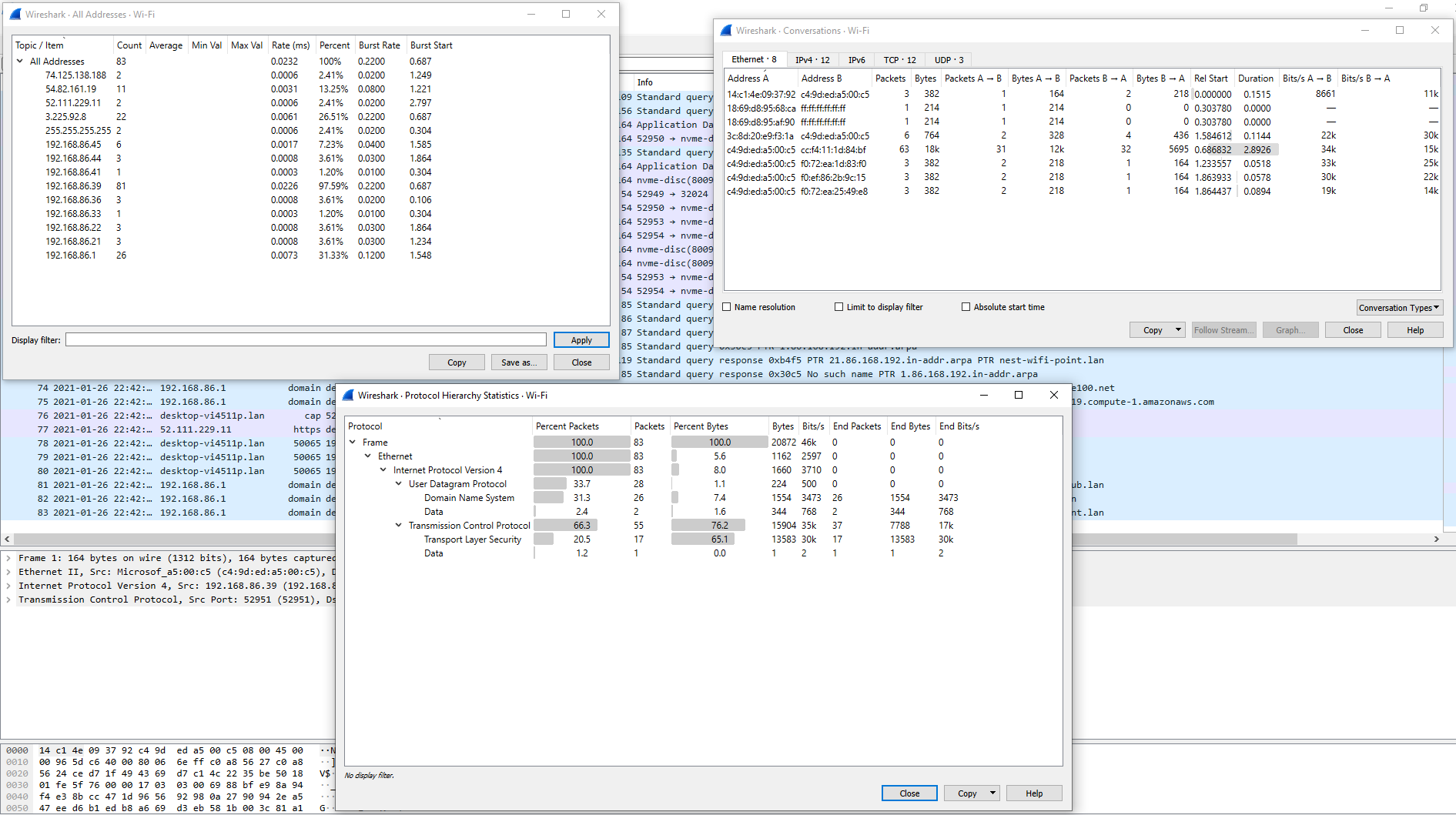
The analyze and statistics radials provide a plethora of plugins to run against the capture. In this section, we will work through a couple of them. We would cover all of which Wireshark offers, but sadly, it is simply not achievable in an introductory module. I urge everyone to experiment and play as we go through this journey.

**The Statistics and Analyze Tabs**

The Statistics and Analyze tabs can provide us with great insight into the data we are examining. From these points, we can utilize many of the baked-in plugins Wireshark has to offer.

The plugins here can give us detailed reports about the network traffic being utilized. It can show us everything from the top talkers in our environment to specific conversations and even breakdown by IP and protocol.

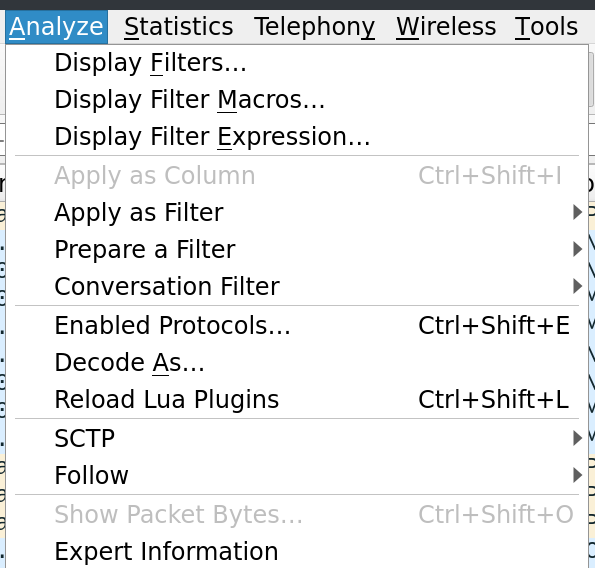
**Statistics Tab**



**Analyze**

From the Analyze tab, we can utilize plugins that allow us to do things such as following TCP streams, filter on conversation types, prepare new packet filters and examine the expert info Wireshark generates about the traffic. Below are a few examples of how to use these plugins.

**Analyze Tab**



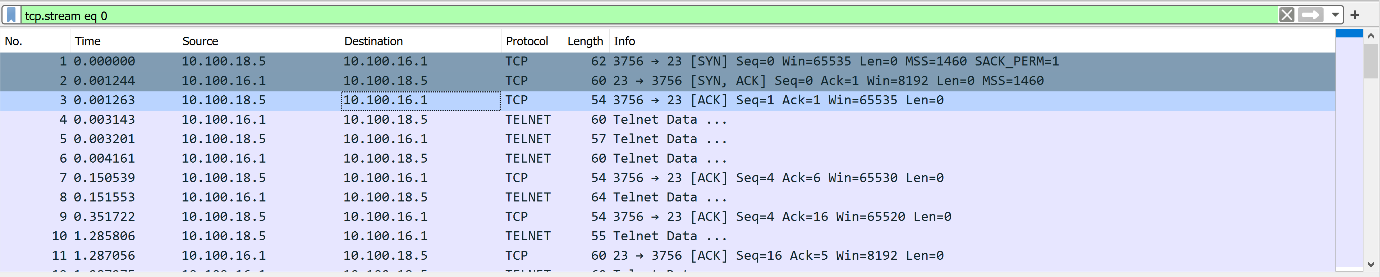
**Following TCP Streams**

Wireshark can stitch TCP packets back together to recreate the entire stream in a readable format. This ability also allows us to pull data (images, files, etc.) out of the capture. This works for almost any protocol that utilizes TCP as a transport mechanism.

To utilize this feature:

* right-click on a packet from the stream we wish to recreate.
* select follow → TCP
* this will open a new window with the stream stitched back together. From here, we can see the entire conversation.

**Filter For A Specific TCP Stream**



Notice that the first three packets in the image above have a full TCP handshake. Following those packets, we can see the stream transferring data. We have cleared anything not related out of view by utilizing the filter, and we now can see the conversation in order.

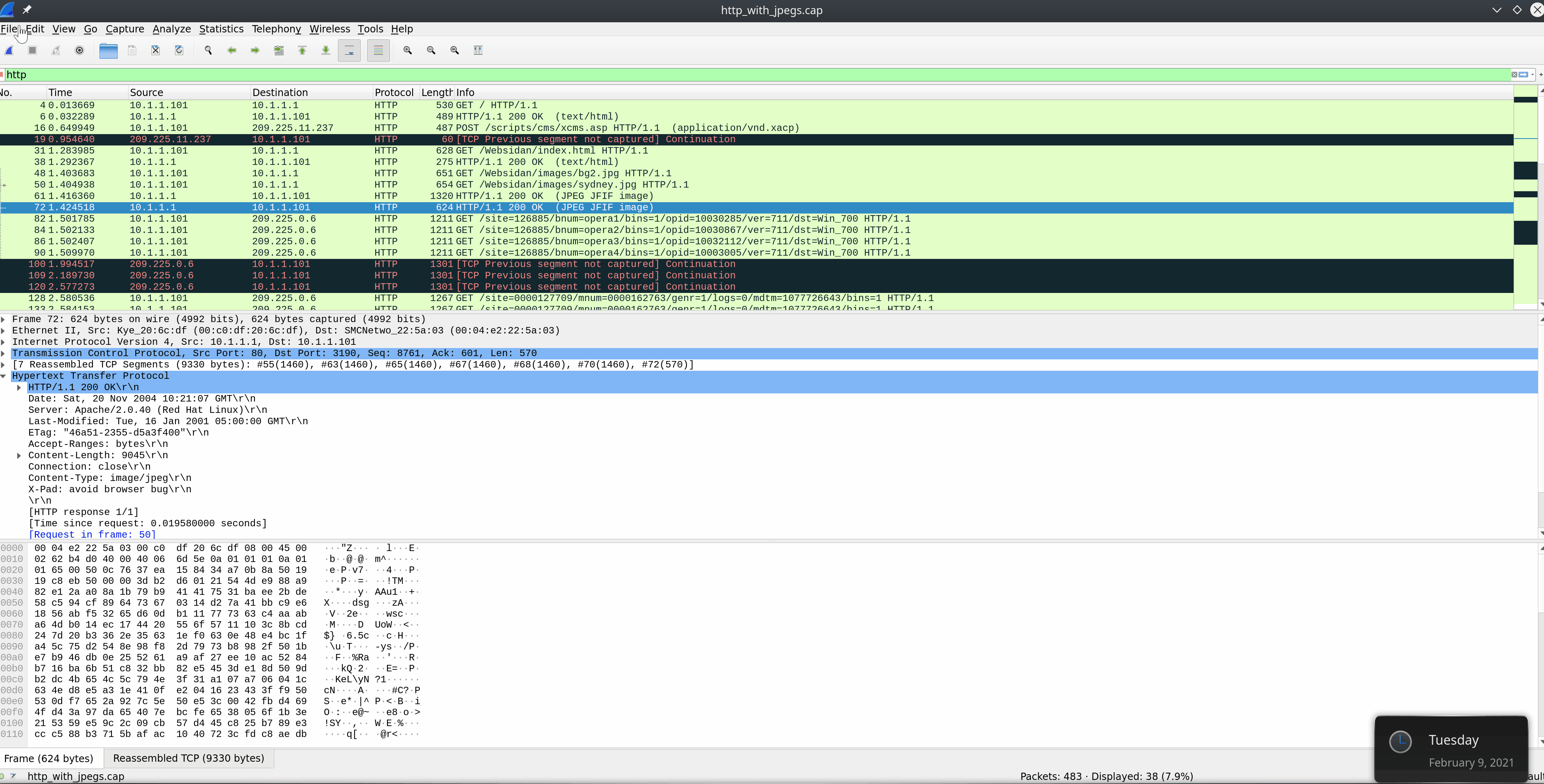
**Extracting Data and Files From a Capture**

Wireshark can recover many different types of data from streams. It requires you to have captured the entire conversation. Otherwise, this ability will fail to put an incomplete datagram back together. If we want a more in-depth understanding of how this capability works, check out the Networking 101 Module or research TCP/IP fragmentation.

To extract files from a stream:

* stop your capture.
* Select the File radial → Export → , then select the protocol format to extract from.
* (DICOM, HTTP, SMB, etc.)

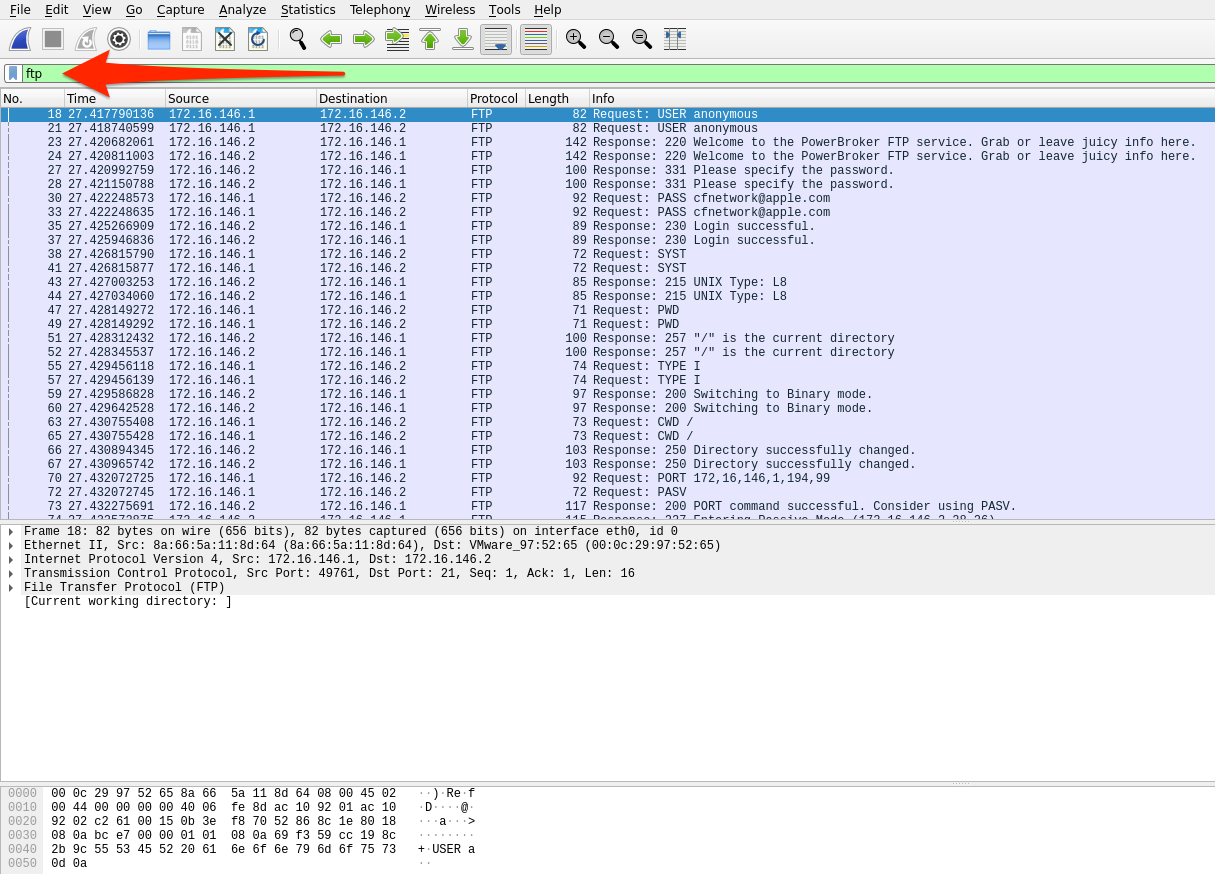
**Extract Files From The GUI**



Another exciting way to grab data out of the pcap file comes from FTP. The File Transfer Protocol moves data between a server and host to pull it out of the raw bytes and reconstruct the file. (image, text documents, etc.) FTP utilizes TCP as its transport protocol and uses ports 20 & 21 to function. TCP port 20 is used to transfer data between the server and host, while port 21 is used as the FTP control port. Any commands such as login, listing files, and issuing download or uploads happen over this port. To do so, we need to look at the different FTP display filters in Wireshark. A complete list of these can be found [here](https://www.wireshark.org/docs/dfref/f/ftp.html). For now, we will look at three:

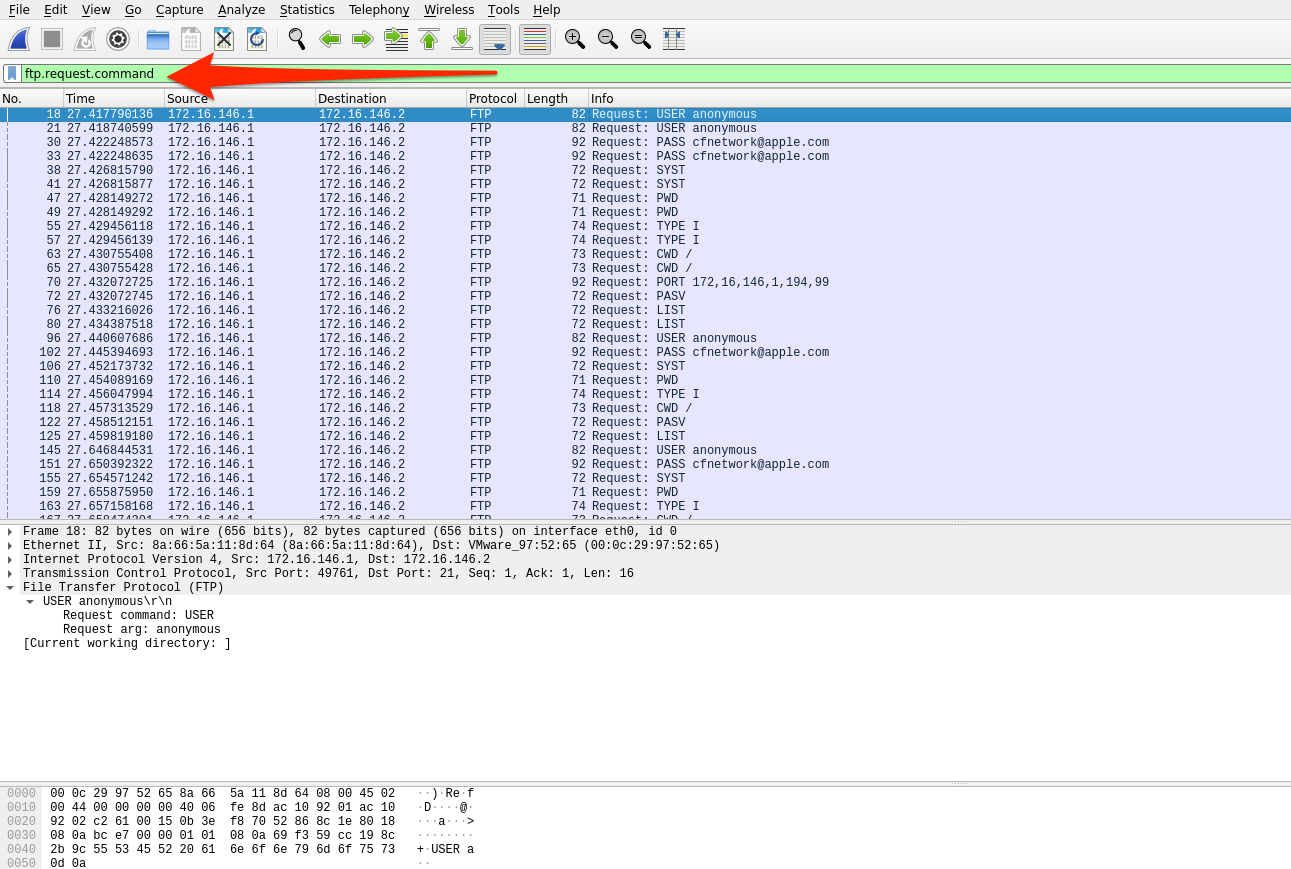
* ftp - Will display anything about the FTP protocol.
  + We can utilize this to get a feel for what hosts/servers are transferring data over FTP.

**FTP Disector**



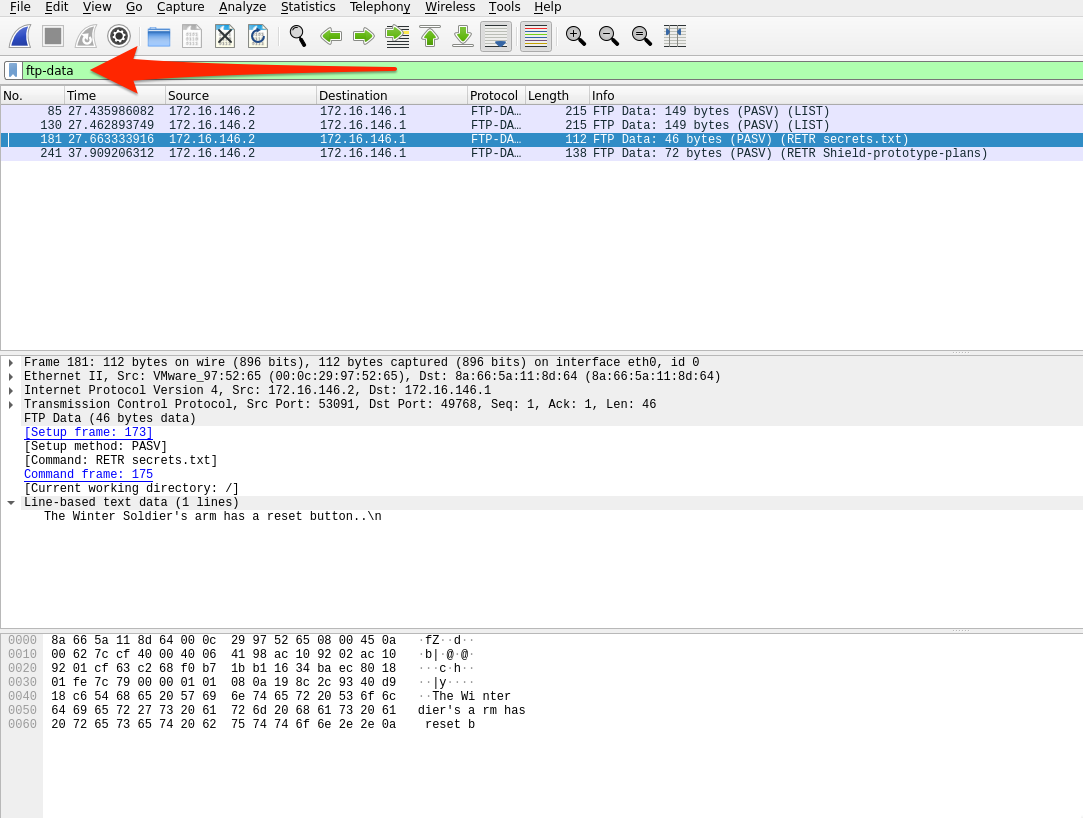
* ftp.request.command - Will show any commands sent across the ftp-control channel ( port 21 )
  + We can look for information like usernames and passwords with this filter. It can also show us filenames for anything requested.

**FTP-Request-Command Filter**



* ftp-data - Will show any data transferred over the data channel ( port 20 )
  + If we filter on a conversation and utilize ftp-data, we can capture anything sent during the conversation. We can reconstruct anything transferred by placing the raw data back into a new file and naming it appropriately.

**FTP-Data Filter**



Since FTP utilizes TCP as its transport mechanism, we can utilize the follow tcp stream function we utilized earlier in the section to group any conversation we wish to explore. The basic steps to dissect FTP data from a pcap are as follows:

1. Identify any FTP traffic using the ftp display filter.
2. Look at the command controls sent between the server and hosts to determine if anything was transferred and who did so with the ftp.request.command filter.
3. Choose a file, then filter for ftp-data. Select a packet that corresponds with our file of interest and follow the TCP stream that correlates to it.
4. Once done, Change "Show and save data as" to "Raw" and save the content as the original file name.
5. Validate the extraction by checking the file type.