**Network Sniffing and Packet Analysis Exercise**

**What is packet analysis and how to capture network traffic?**

A **packet analyzer** is a piece of computer hardware or software that can intercept and log traffic passing over a digital network. When a network request is made (i.e. a web-page search, sent email message…) information is sent across the network from the source location to the destination via multiple data streams/packets. These streams contain header information that describes among other things the source of the request, the destination of the request, the type of data contained in the packet, various information describing the transaction and is then followed by the actual data. On simple web search can generate many data packets.

In this exercise we will analyze some previously captured traffic and explain the contents of the data in detail. We will discuss how to filter captured data streams to limit simplify and fine tune analysis. Through these demonstrations we will enlighten you on safety procedures and risks involved in running certain applications. Specifically we will analyze the following types of network traffic:

* Web Server – http
* File transfer Protocol – FTP
* Port scan
* Password cracking attempt

For these exercises we will use the Wireshark software application that has been installed on your virtual machine. Wireshark, formerly known as Ethereal, is a very powerful tool for network analysis. Wireshark is especially popular because it runs on Windows, Mac OS and Linux. It is a network packet analyzer that can peer inside the network and examine the details of traffic at varying levels. The information it can show you range from application-level information to the actual bits in a single packet.

We will analyze network traffic that was previously captured and has also been installed on your machine. You will need to connect to the vSphere Web Client. Start up and log in to the Snapshot entitled Packet Sniffing Exercise.

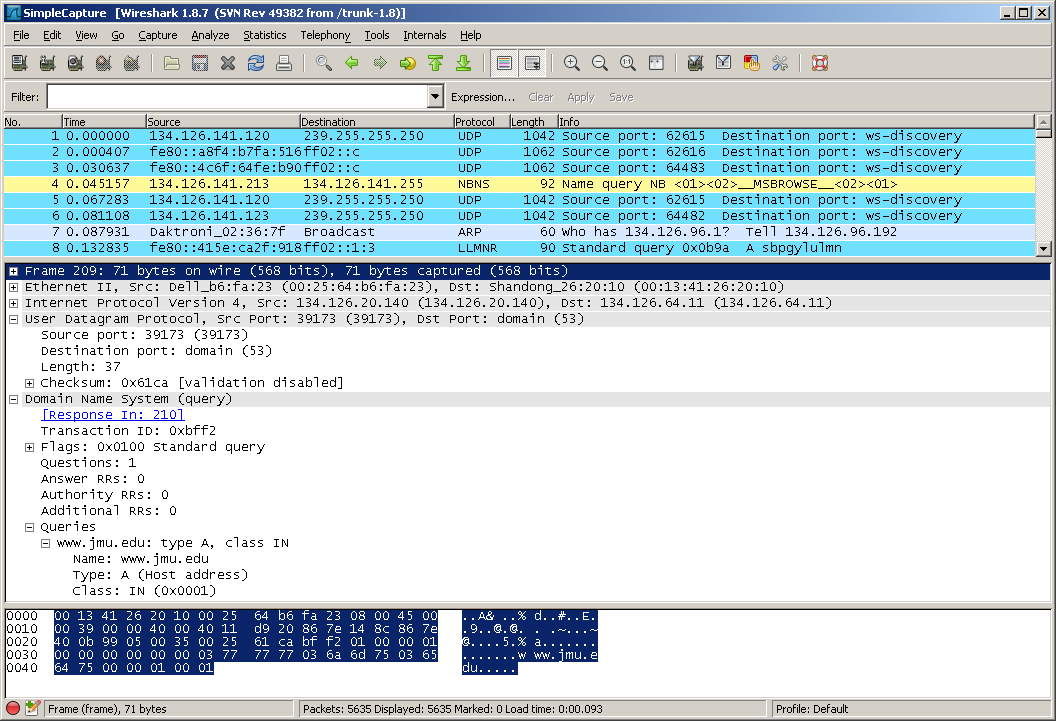
**Exercise 1 –Using Wireshark to analyze basic network traffic**

We will begin by analyzing a simple network traffic capture. *Double-click* on the ***Wireshark*** desktop icon. When Wireshark is used to capture and save network traffic it is saved in format known as a ***.pcap*** (packet capture) file.

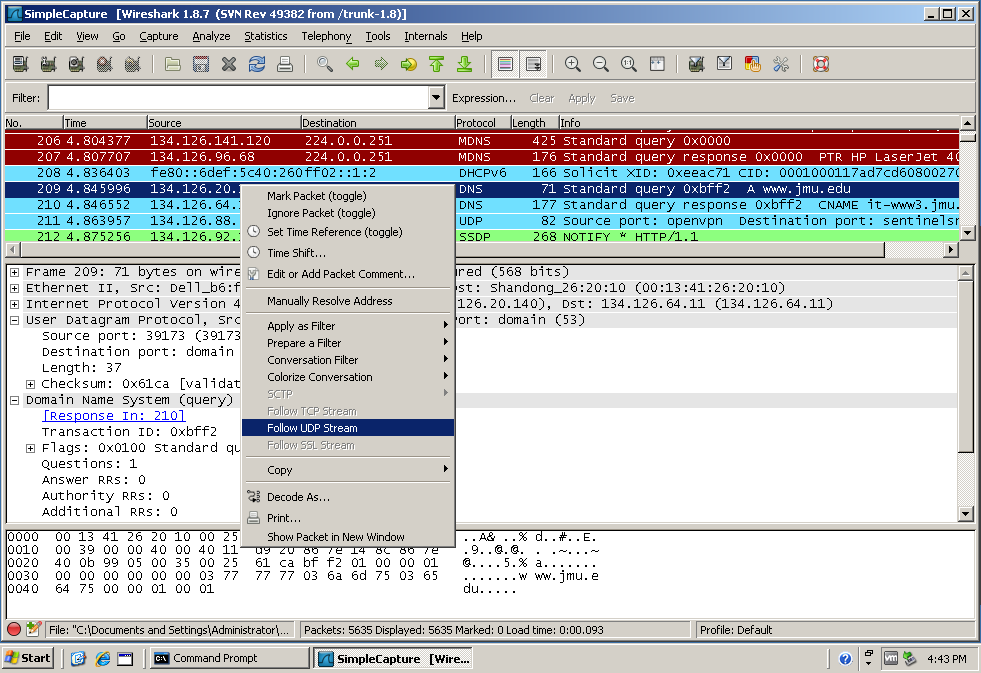
Click on *File → Open* and select the **SimpleCapture.pcap** file located on your desktop.

**Wireshark General Layout**

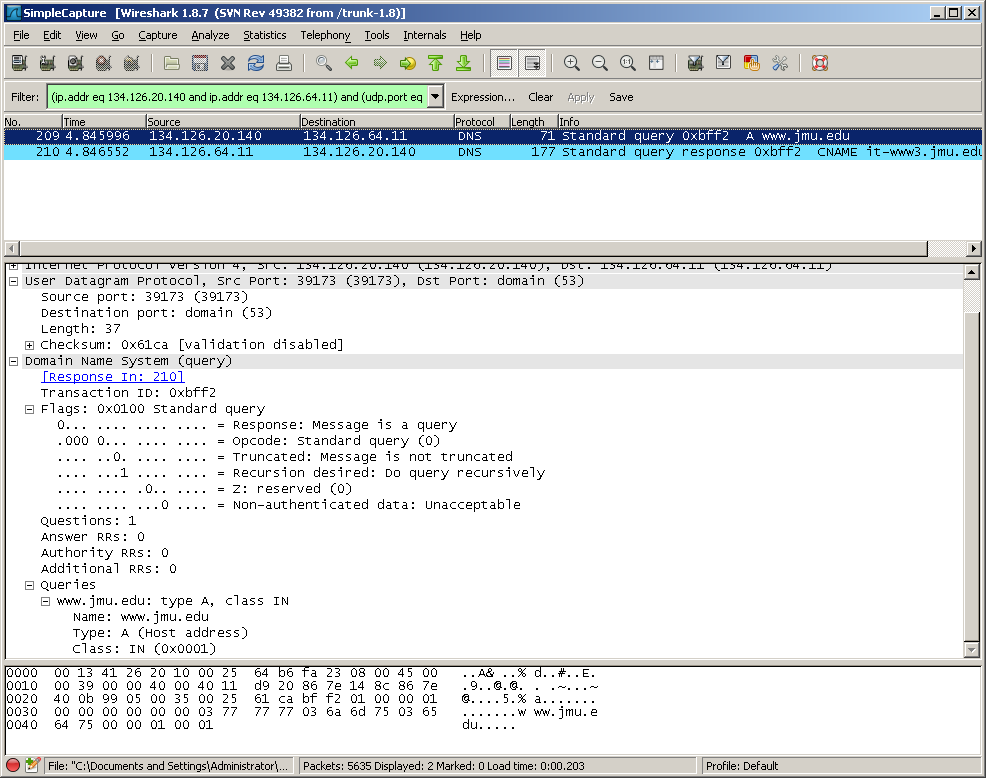
Each line in the capture corresponds to a single packet seen on the network. This is shown in the top pane. The default display shows the time of the packet (relative to the start of the capture) as well as the source and destination IP addresses, the protocol used and some information about the packet. You can click on a row to obtain more information. This allows the other windows to be used. The middle pane contains more internal details on the packet selected in the top frame. These can be expanded out into varying levels of detail. The bottom screen displays the actual data. On the left-hand side you see the hexadecimal representation of the data. On the right-hand side the character representation is displayed. Note the headings displayed in the first section. The first column denotes the packet number. The second column is the time relative to the start of the capture. The remaining columns are the Source IP address, the Destination IP address, the Network Protocol, the Packet Length and Information about the packet.

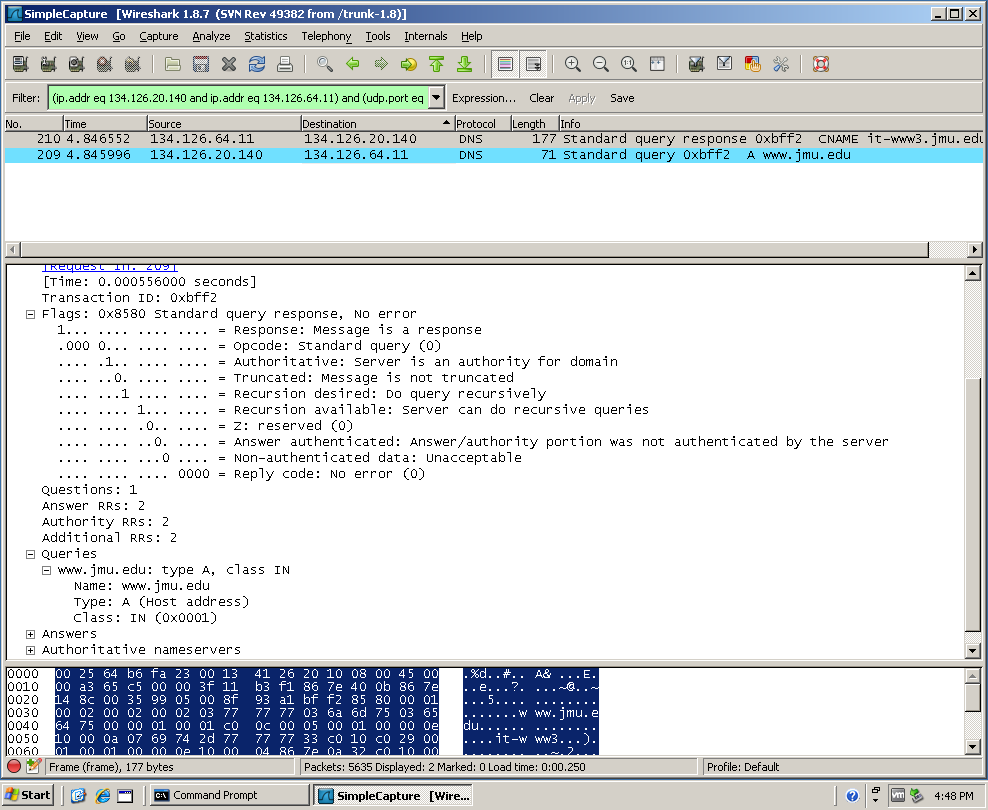


We will begin by examining the sequence of events that take place when a user performs a DNS query. DNS stands for *D*omain *N*ame *S*ervices. DNS takes a common fully qualified domain name and translates it to a corresponding Internet address. The sequences of events that take place are first a request is made from a source machine to a DNS server. If the server recognizes the name requested it sends a response with the IP address associated with the name. Two possible situations can occur if the server does not know the name requested. These situations depend upon how the request is configured. If the request is a recursive request then the source machine will depend on the server to forward on the request to find the answer. If the request is setup to be iterative then the server will respond that it does not know the name and the source machine will need to make a request to another server. Let’s take a look at a simple request that was made in our network capture. In our example a user has made a request via the *nslookup* command to get the ip address for ***www.google.com.*** To view this traffic in Wireshark, scroll down and select packet # ***304***. *Right-click* on the packet and select *Follow UDP Stream*

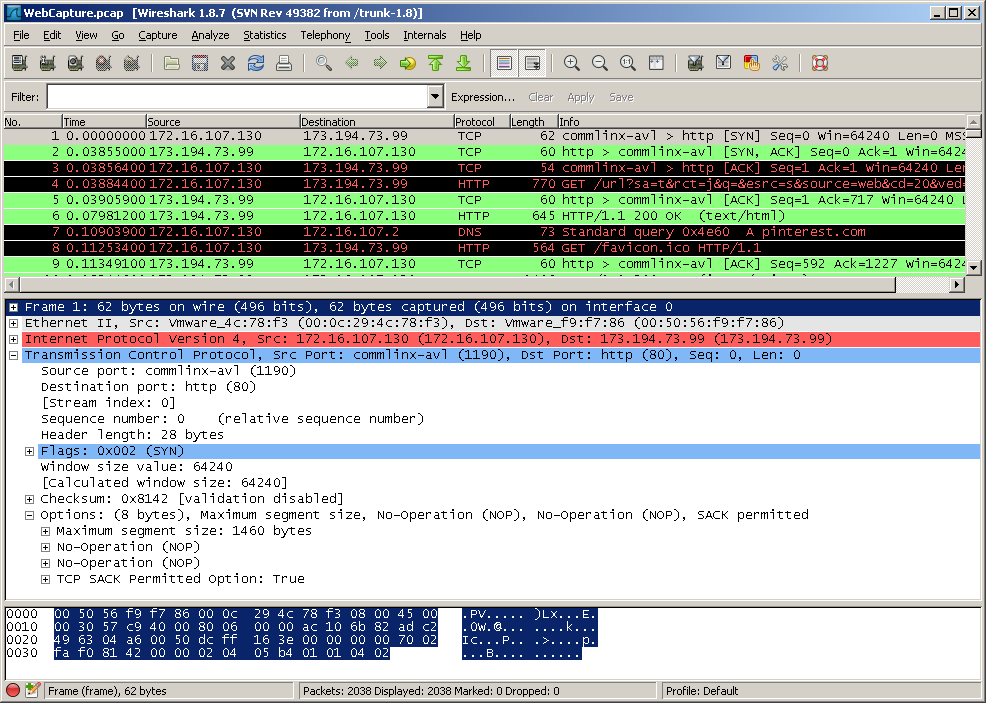
******

A Follow UDP Stream will appear. You will notice that the Stream Content contains www.google.com and other nonsensical characters. Close this window and return to the main Wireshark window. Notice now that only two packets appear. Click on the first packet and look in the second pane. If you look in the flags section you will notice that this is a recursive query. This tells us that the server will send on the request if it does not have an answer.





A *three-way handshake* is used to initiate communication between two machines. When a source machine wants to communicate with a destination machine it will start by sending a SYN request. This tells the destination machine that a conversation is being requested. If the destination machine accepts the request it will respond with a SYN, ACK. When the initial machine receives this response it in turn responds with an ACK response and the conversation begins. Lets close our current packet capture and open another to observe this traffic. Go to *File 🡪 Close* to close the current capture. Then choose *File 🡪 Open* and select the file named **WebCapture.pcap**.

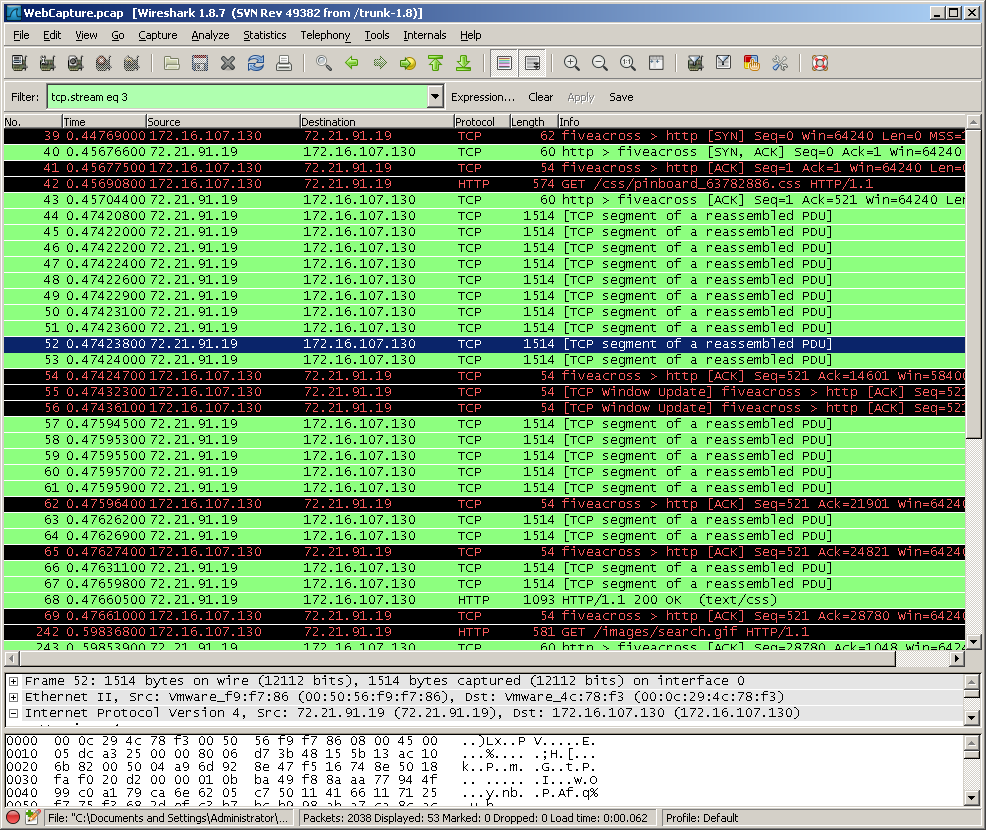


This is a capture of a simple web search. The user has opened up the Google search engine and ran a search for “cute puppy pictures”. The user then chose the Pinterest webpage, selected and downloaded a picture.

Lets take a look at the packet capture in detail. In the first packet we see that the source computer (172.16.107.130) sends a SYN request to the destination computer (173.194.73.99). If you expand the Transmission Control Protocol setion of the second pane you can see that the source port is 1190 and the destination port is 80 (indicating an http request is coming). The Flags section shows that this is an initial SYN request. The next packet displays the SYN,ACK response coming back from 173.194.73.99 port 80 to 172.16.107.130 port 1190. Packet 3 shows the final ACK response completing the three-way handshake.

Packet 4 marks the beginning of the “cute puppy pictures” search. Packet 7 shows the DNS lookup (query) for pinterest.com (the site where the puppy picture resides). The DNS response is seen in packet 11. Packets 12-17 indicate two different three-way handshakes for requests from pinterest.

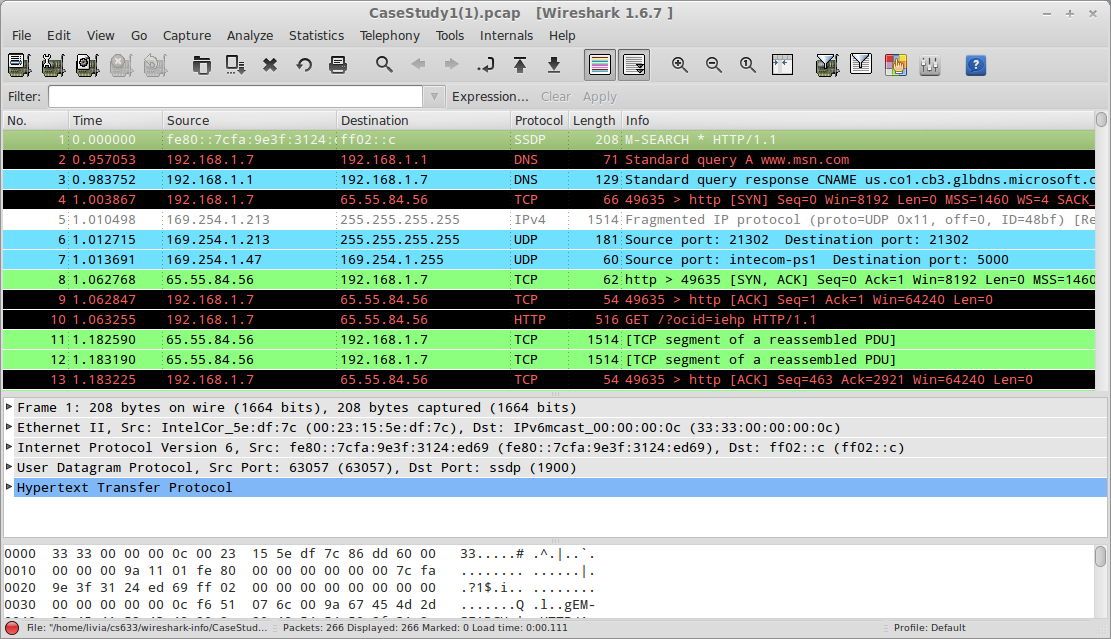
Scroll down and click on packet 42. Right click and choose Follow TCP Stream. Notice the two sets of headers followed by a bunch of confusing information. This confusing information is the binary picture being downloaded. If you close the Follow TCP Stream window you will notice a filter has been entered in the filter section (we will talk more about filters later). What is important to know at this point is that this request has filtered out all other packets and now we can follow just this network conversation.



This section basically shows the downloading of the image. Notice all the lines which contain “15114 [TCP segment of a reassembled PDU]”. This is portions of the image being downloaded.

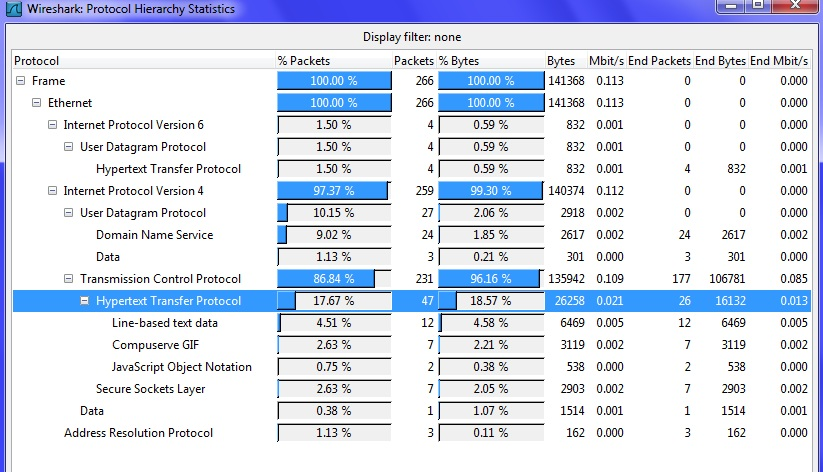
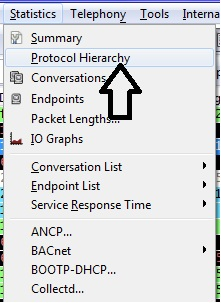
This concludes this exercise close the Capture file and procede to Exercise 2.

**Exercise 2** - **Gathering General Information and Statistics and More Wireshark Analysis**

****Click on *File → Open* and select the **CaseStudy1.pcap** file located on your desktop. You should now have a window displaying that is similar to below:

To determine what type of data has been captured in this file we can to go the *Statistics* section and select

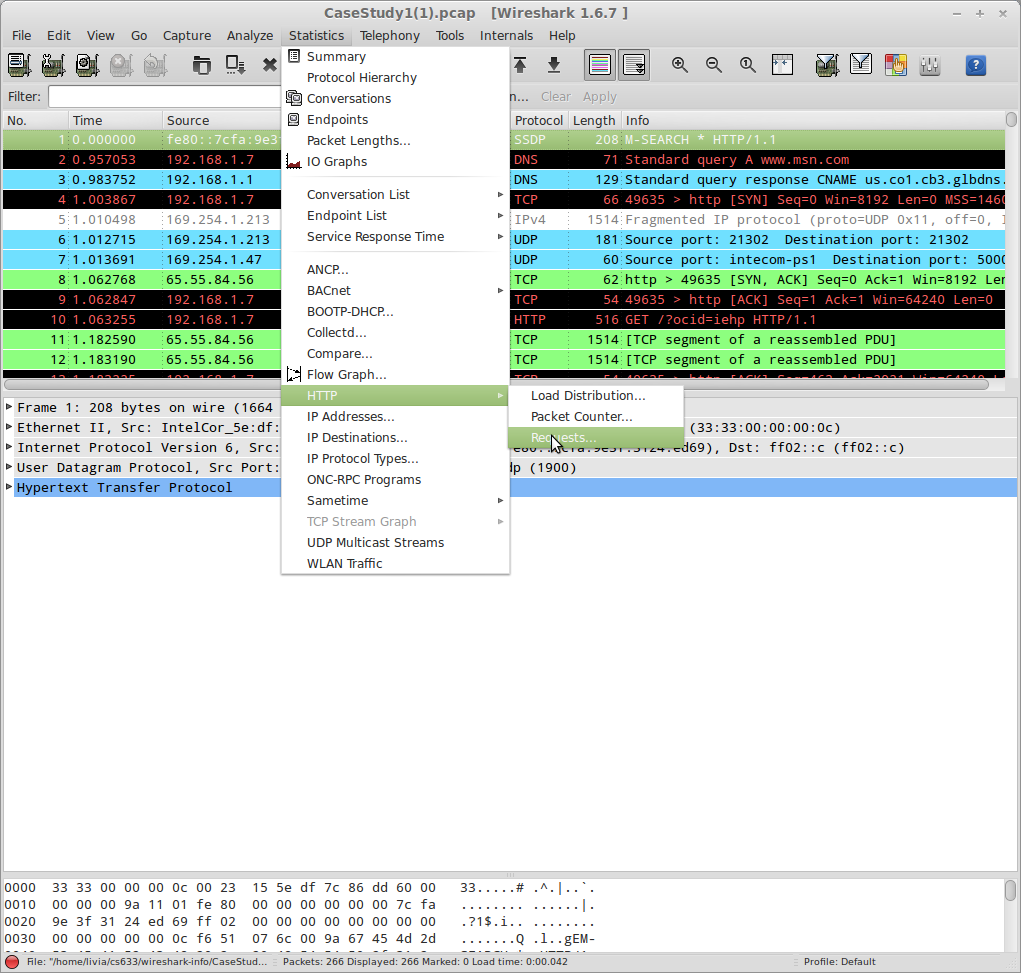
*Protocol Hierarchy*

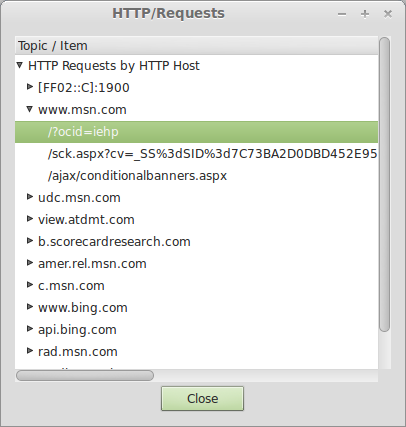
****

Notice the line highlighted above. This shows that 17.67% of the traffic contained in this capture is Hypertext Transfer protocol (http – web traffic). This may not seem like a lot but you must remember that one web request will generate several packets of data.

Other interesting statistics can be gathered easily through selections made under the *Statistics* section. We will examine a few briefly.

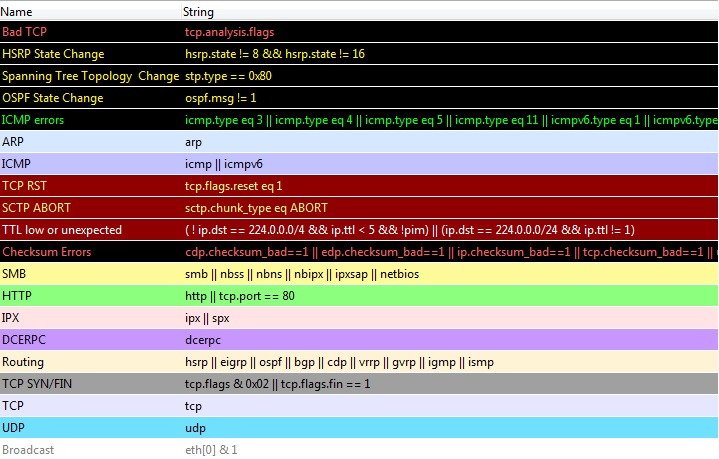
Close your current window and select the Statistics section again. This time choose *HTTP 🡪 Requests…* Leave the filter blank and click on Create Stat



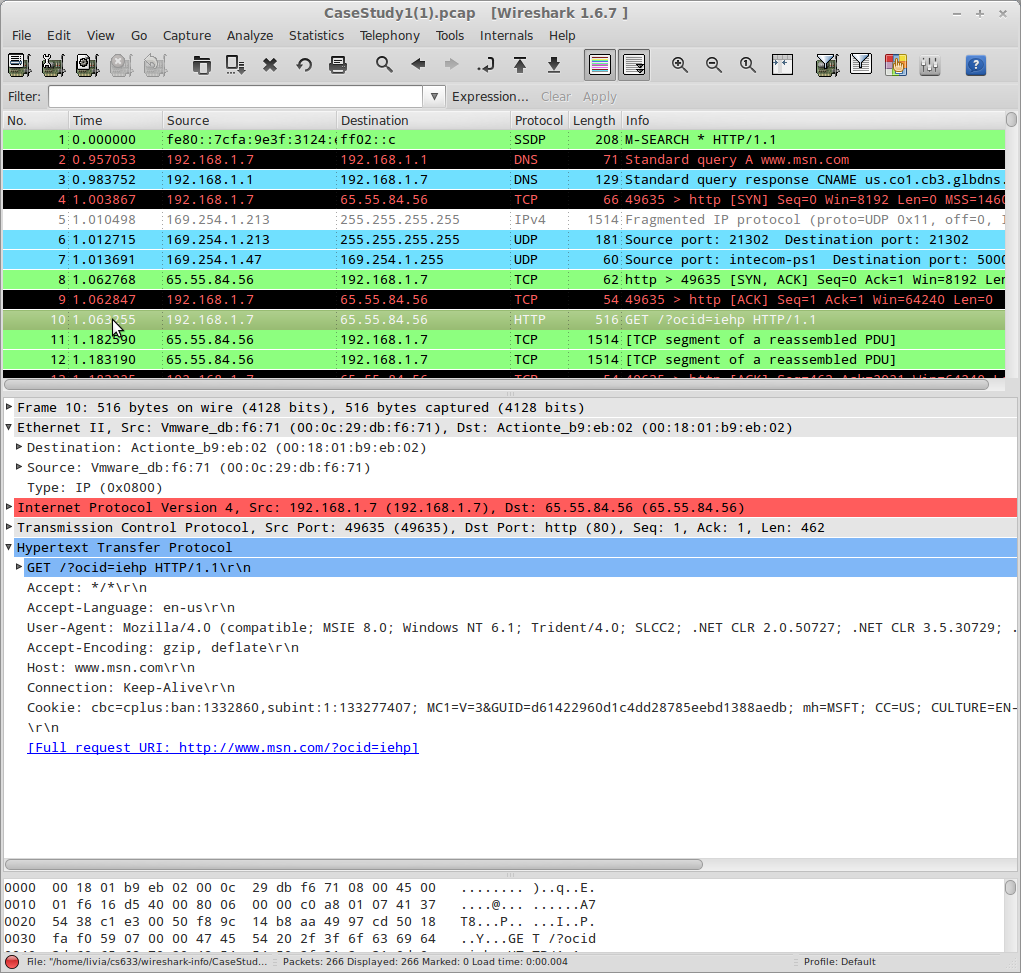


(The above figure might be obsolete and what you see might be a little bit different)

This list now contains all the http requests made.

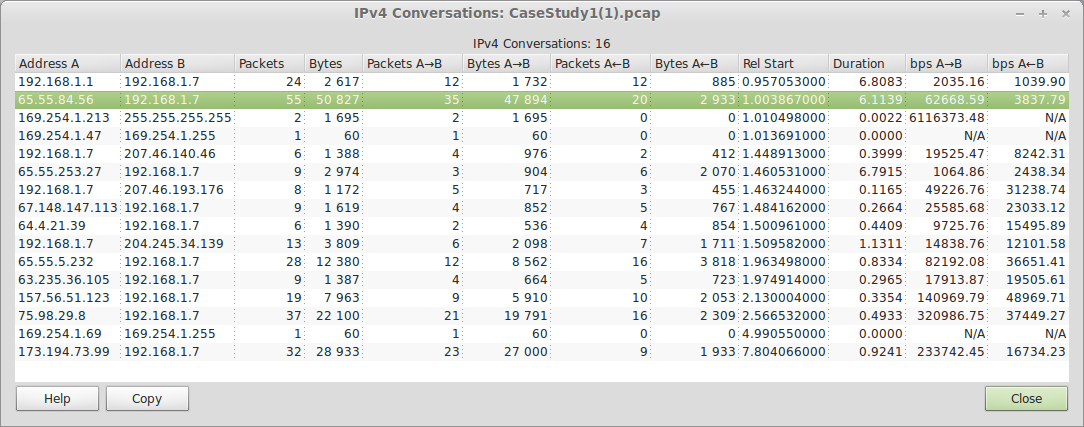
Wireshark includes a complex color-coding scheme. The default settings are as follows:

Close this window and lets analyze the first http packet – click on line 10.

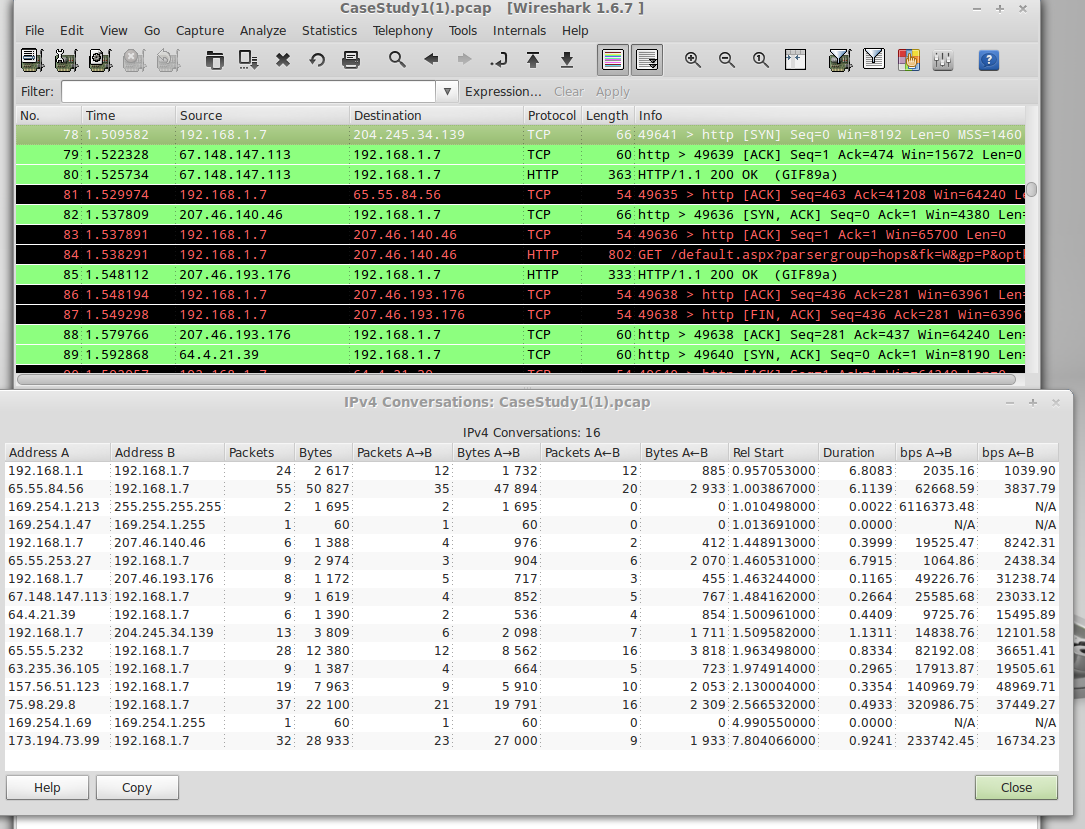


Examine the information displayed in the middle pane of the windows. Wireshark does a great job of showing all the details of the packet (including the lower levels and not just the application layer). In this case, we get the most information from the application layer. Here we can tell that the HTTP packet is a GET request for the /?ocid=iehp page of [www.msn.com](http://www.msn.com/). Wireshark even summarizes the full request below to be http://www.msn.com/?ocid=iehp.

Conversations

A network conversation is the traffic between two specific endpoints. Along with the addresses, packet counters, and byte counters, this window also has the time in seconds between the start of the capture and the start of the conversation (“Rel Start”), the duration of the conversation in seconds, and the average bits (not bytes) per second in each direction. Lets take a look. Click on the Statistics section and go to Conversation list → IPv4. A window similar to that below should appear.

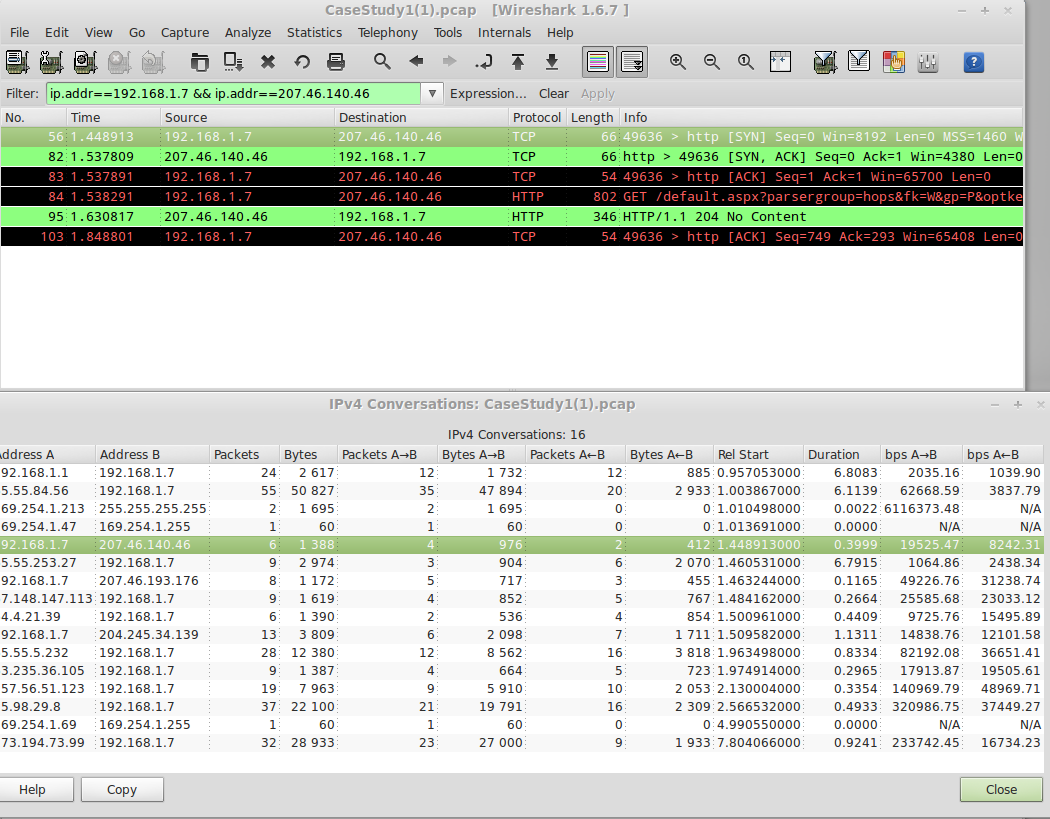
Each row in the list shows the statistical values for exactly one conversation. Conversations can be further shown at each of the different levels. From the conversation window, filters can also be applied. To demonstrate this situate this window so that it is so that you can see both it and the top pane of the main Wireshark window as is demonstrated below:



Since we have created no filters all the network traffic appears. Choose the fifth line down in the conversation window

A description...

Right-click on the line and choose Apply Filter then Selected then A ↔ B. Notice how the contents of the first pane of the main Wireshark Window has changed. Now you are viewing only the traffic which transpired between the source IP address of 192.168.1.7 and the destination address of 207.46.140.46



Notice that the Filter section above the first pane has been filled in. Close the Conversation window and lets examine this further. The Filter that we applied was:

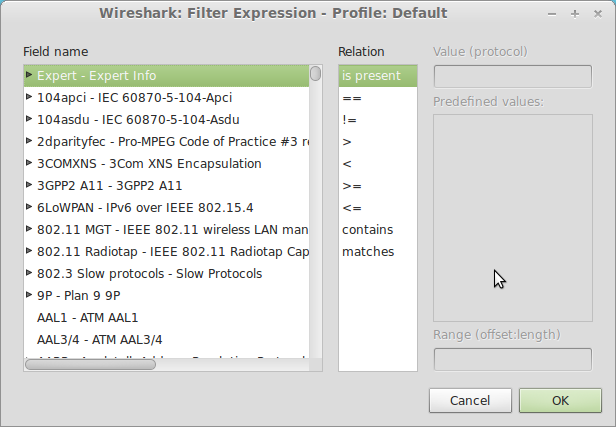
A description...

This was one easy way to filter out all traffic except that between IP address 192.168.1.7 and IP address 207.46.140.46.

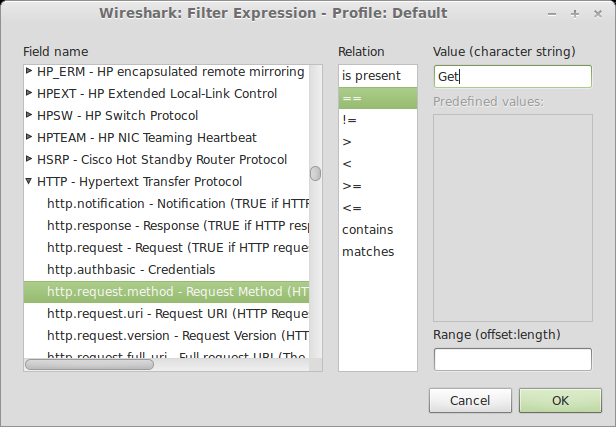
Filters are a good way to decipher through all the packets and zone in on specific information. Let's examine a few simple filters that we can make through the use of the *Expression...* builder. Press the *Clear* button located to the Filter and then press the *Expression...* key.

A description...

It may take a few seconds but a Filter Expression window should appear



Here you can build your own filters. Scroll down to ***Http*** in the *Field Name* section and expand the options. Notice that there are many different sub-filters that you can use to fine tune your search. Lets narrow our search to all the Get requests. Click on the sub-filter labeled *http.request.method.* In the Relation section choose the = = “equal” sign. Notice that the Value section becomes active. Type in **GET** (be sure to use all CAPITAL Letters)and click *OK*



Notice that the filter has been filled in:

A description...

Click on Apply and notice the packets that have been selected. This is a list of all the Get requests that have been made in the captured session.

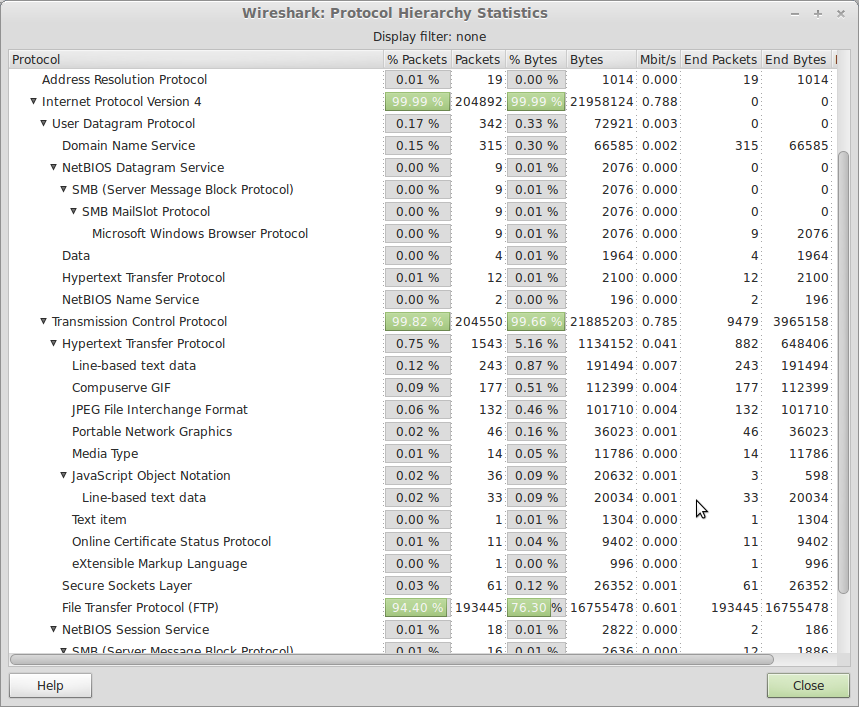
This ends the first exercise. Click *Clear* on the Filter line to exist the filter. Then go to the *File* tab and select *Close* to close the ***CaseStudy1.pcap*** file.

**Exercise 2 – More Traffic searches and Filters**

Go to the *File → Open* tab and open the file located on your desktop named ***CaseStudy2.pcap***

In this packet capture you will immediately notice that there is a lot more traffic. Lets start by looking at the *Protocol Hierarchy* statistics to see if we can gather some information about what events have taken place. If you remember from the previous exercise this can be done by selecting *Statistics → Protocol Hierarchy*

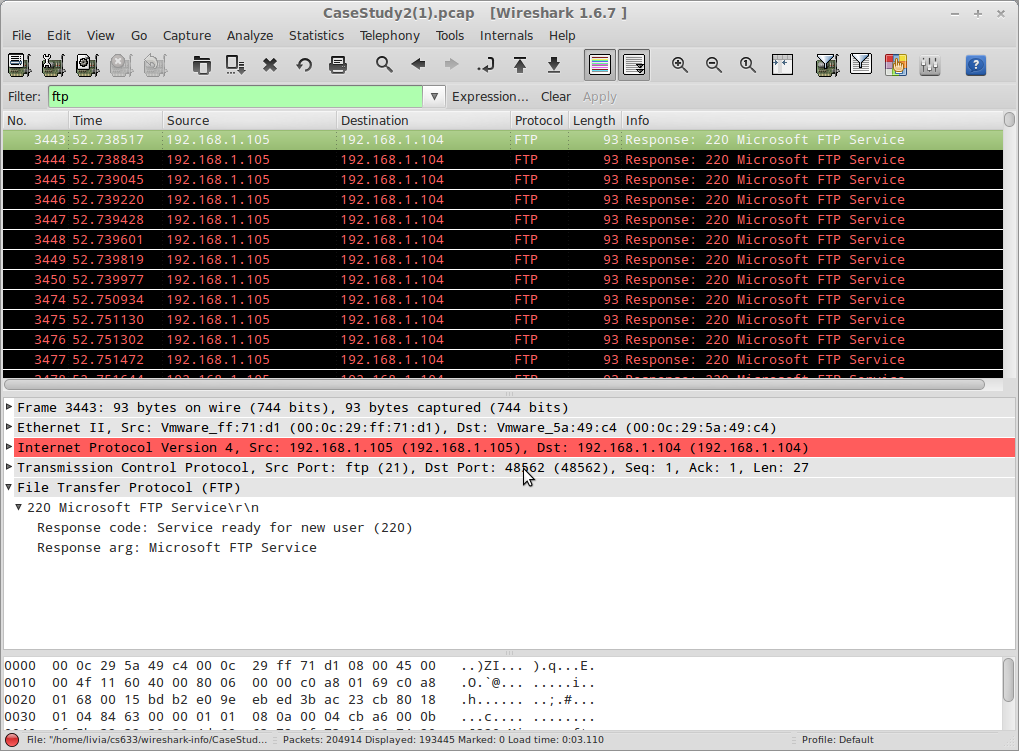
If you enlarge the window you will notice that **94.4%** of the Packet traffic took involves the **File Transfer Protocol (FTP)**



This amount of FTP traffic in one network traffic seems very suspicious and warrants closer examination. To do this we need to apply a filter. This can be simply done by typing the word ftp in the Filter Section of the main Window and clicking apply

A description...

This traffic appears even more suspicious since at first glance it appears that there are multiple repeated packets. This does not resemble the expected traffic of a legitimate FTP request. Lets examine some of the packets in more detail. Click on one of the first packets in the Window and fully expand the File Transfer Protocol (FTP) section in the second pane

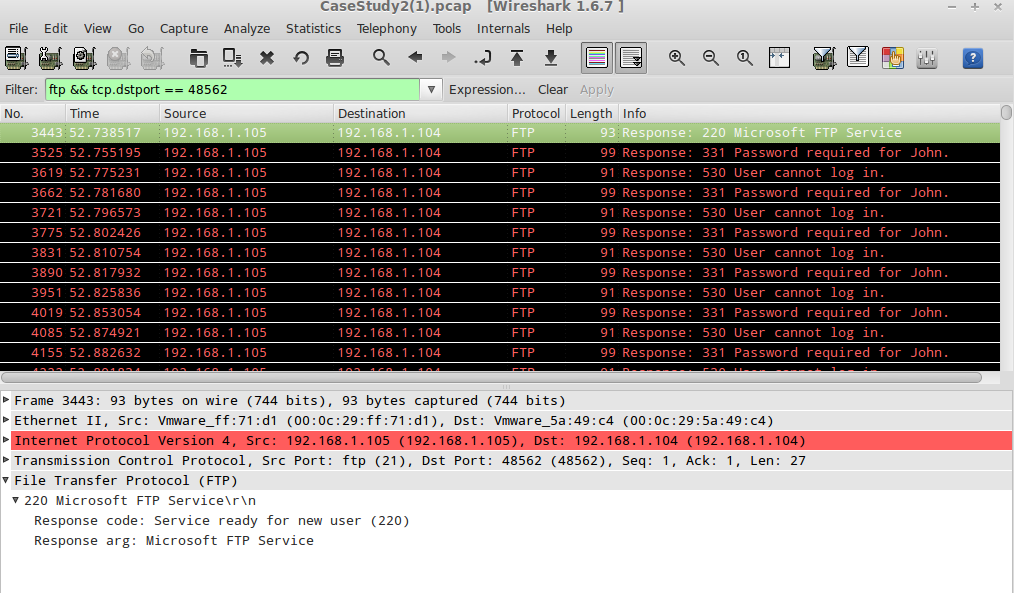


From this we gather that it appears that FTP is ready for a new user. This seems to suggest multiple sessions attempting to be opened and that they are all being attempted on same IP address. Click on the next several packets and take note of the Destination Port. This is pointed to by the the pointer in the above display and appears in the following line

A description...

Click on the next Packet in the first frame and notice that the Destination port has increased from 48562 to 48563. Click on the next several packets and again take note of the destination port's incremental changes. This raises another red flag and warrants further investigation. Let's fine tune our filter and see what we can find out. We will filter out all the traffic attempted at one of the destination ports listed. To display a specific destination port we need to use the tcp.dstport filter. Type in the following in the Filter section and select Apply

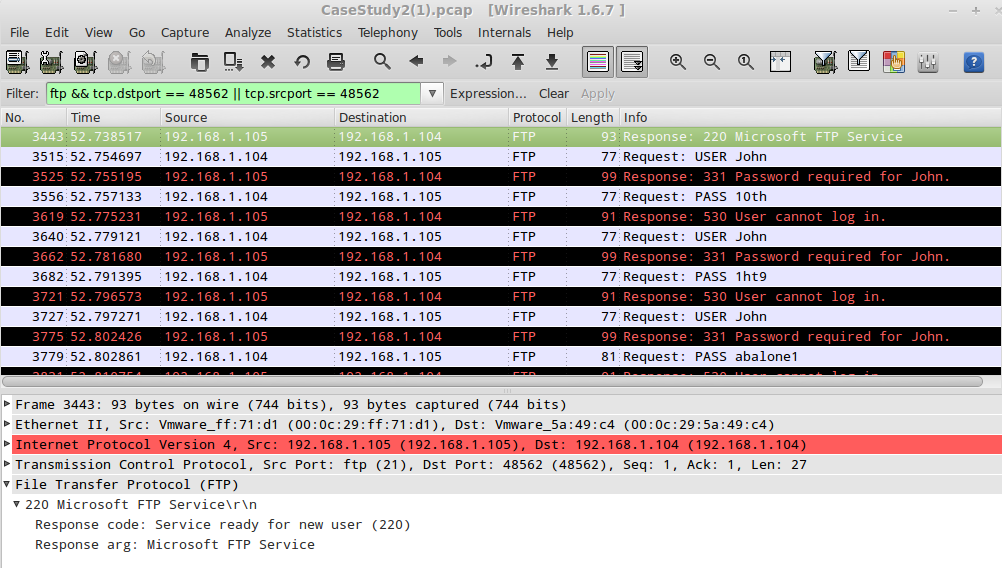
A description...



With this filter it seems that someone is trying to login a multitude of time. We are, however, only getting one side of the conversation. This is because with the tcp.dstport setting we are only seeing the return response and not the original request. We can modify our filter to include both sides of the conversation by adding a filter for the source port as follows:

A description...

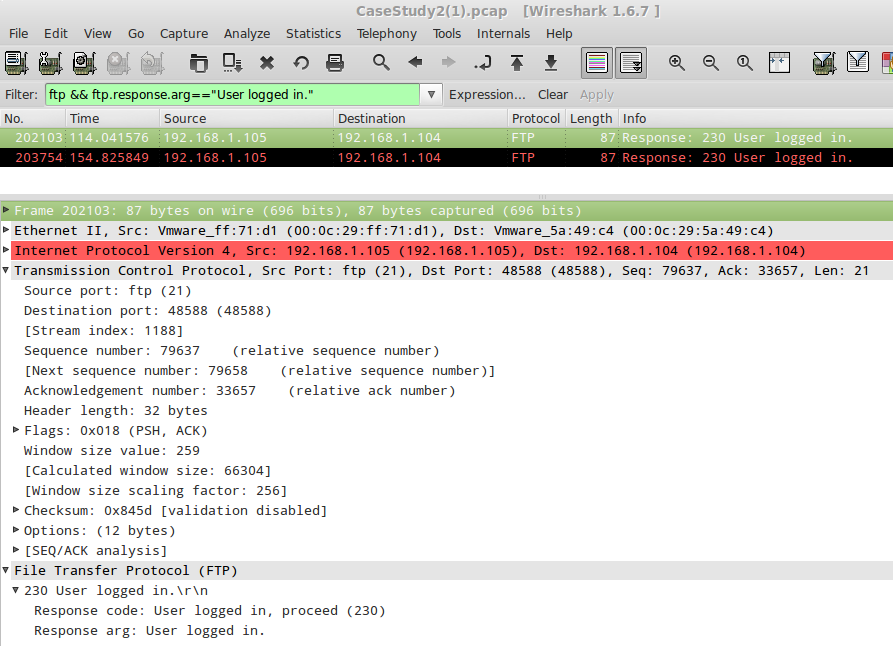
Note the added output:



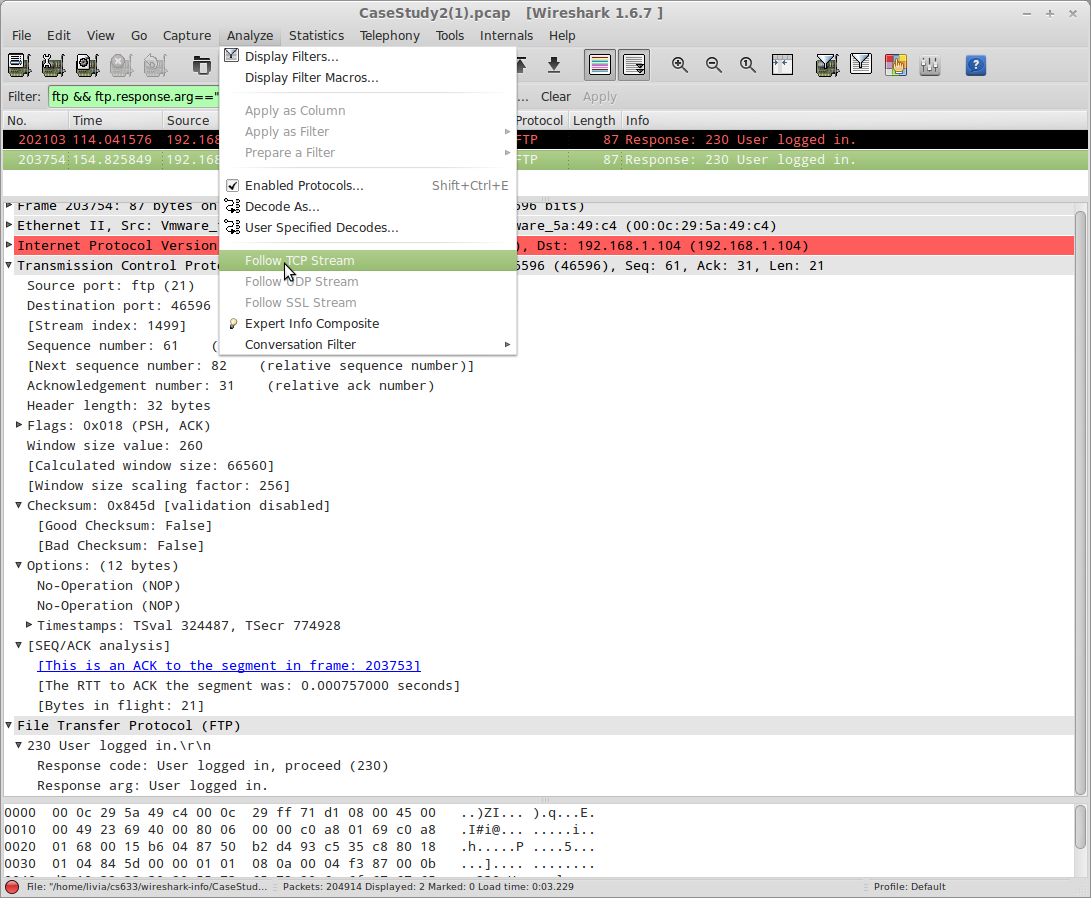
With this information we can finally see what is happening. It appears that this is a password attack against and ftp server. The only thing we do not know at this point is whether the password cracking attack was successful. We can determine this by adding one more filter. When a user successfully logs in the response “User Logged In” is sent. Now we will filter for this response by typing in the filter displayed below (Be sure to type the filter **exactly as it appears** below – including the **period** after the word **in**)

A description...

This filter returns two packets



Examining the detail found in the second pane shows that a successful login was made from IP address 192.168.1.105 to 192.168.1.104 via FTP on Dst Port 48588 and again on Dst Port 46596. To determine what login and password were used we can follow the TCP stream. This can be done by selecting one of the packets. Then go up to the *Analyze* label and click on *Follow TCP Stream*



A Follow TCP Stream window will appear. Scroll down to the last entries and you will see that after multiple unsuccessful attempts the User ***John*** logged in successfully using the password ***Password1234***. John had a very weak password and his account was compromised via the use of a dictionary attack. Beyond this, we could look into the commands issued once the connection was established to determine what the attacker did once he obtained access to FTP.

One question might also be brought up. How did the attacker know FTP was enabled? This might suggest a port scan was performed. To check this, we need to perform a filter on some TCP flags. TCP packets have eight flags. They are FIN, SYN, RST, PSH, ACK, URG, ECE and CWR. These flags have decimal numbers assigned to them as follows:

FIN = 1

SYN = 2

RST = 4

PSH = 8

ACK = 16

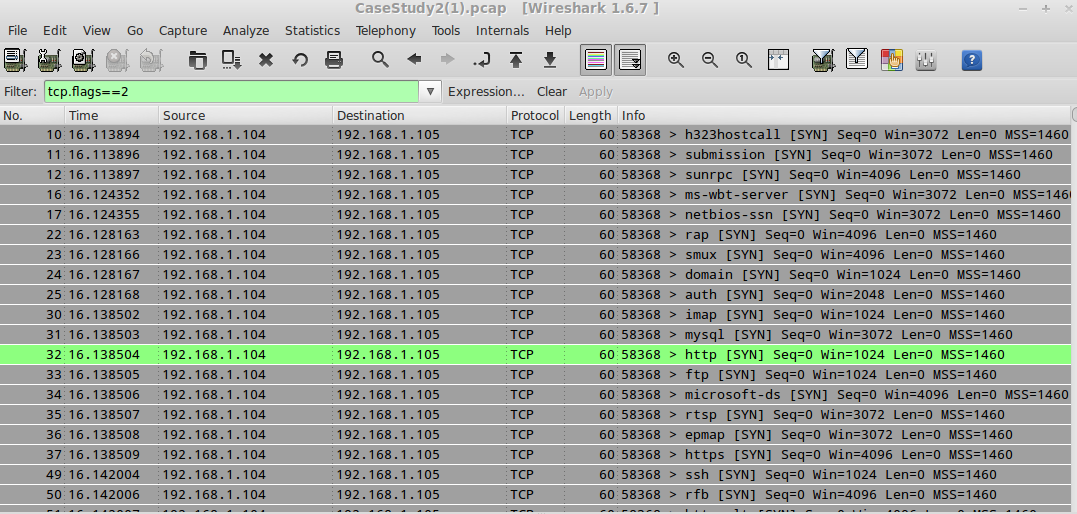
URG = 32

ECE = 64

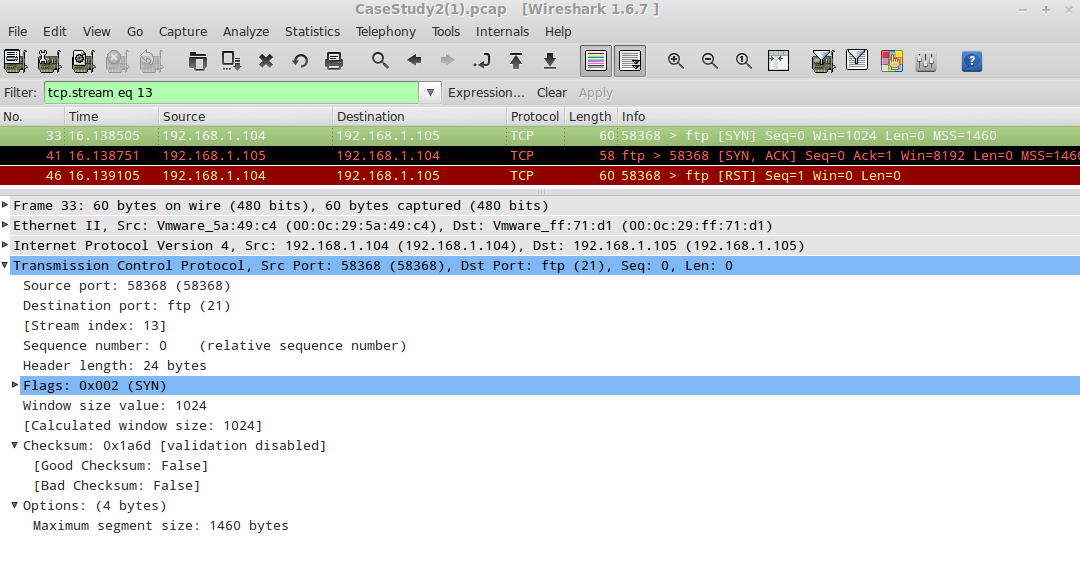
CWR = 129

To check if a SYN/ACK flag is set we add 2 (the SYN value) to 16 (the ACK value) and the result would be 18. A common port scan is a SYN scan, We will first check for that using the following filter:

The resulting output indicates that a SYN scan was indeed executed:



This tells us that the attacker knows the ports that were open and that an FTP server was running. Let's examine a few of these lines to see what ports are open. Since we are already aware that the FTP port is open lets scroll down and select line No 33 (see first column in the first pane). Now go to the *Analyze* tab as select *Follow TCP Stream.* Although nothing appears in the TCP Stream window if we close it we see three packets are displayed.



The first line is Packet Number 33 and is the original SYN resquest. The second line (Packet 41) is the SYN, ACK response. This tells the requester that the port is in fact open. Looking in the second pane we see the Transmission Control Protocol line which confirms that the Dst Port is port 21 (FTP).



Clear the filter and lets try another protocol. Start again by typing tcp.flags == 2 in the filter to filter out all SYN requests. This time select packet number 32 for http, and go to *Analyze → Follow TCP Stream.* Here we see that only two packets appear. The original SYN request and then a RST, ACK response. No SYN, ACK is displayed so can determine that this port is not open and no http server is running (or at least not on port 80).

This concludes the Network Analysis exercise. Please close out of the Wireshark application and shutdown your Network Analysis Virtual Machine.

References:

**Wireshark Case Study[1,2].pdf** presented by Florian Buchholz and Brett Tjaden

Wireshark User Guide

<http://www.wireshark.org/docs/wsug_html_chunked/index.html>

Advanced Wireshark tutorial: Packet and network security analysis

<http://searchsecurity.techtarget.in/tip/Advanced-Wireshark-tutorial-Packet-and-netowrk-security-analysis>

Quick and Dirty Wireshark Tutorial

<http://searchsecurity.techtarget.in/tutorial/Quick-and-dirty-Wireshark-tutorial>