Wireshark for Basic Network Security Analysis

This project will contain details on how to capture network traffic that is flowing through a machine now and analyzing already captured network traffic by opening a stored capture file in Wireshark. It will also contain the differences between Wireshark Capture and Display Filters. Generating, capturing, and analyzing RADIUS, HTTP, DNS and Telnet Traffic. It will contain Decrypting RADIUS passwords. Generating, capturing analyzing encrypted protocols such as SSH and HTTPS and decrypt HTTPS traffic

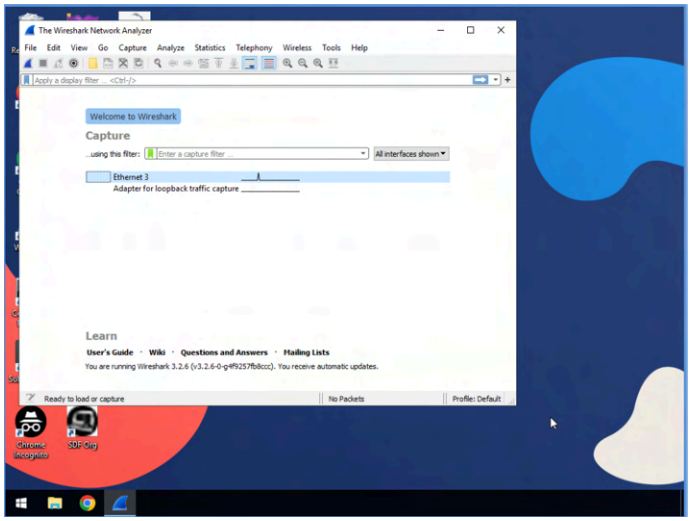
Scope and Objectives

This project will include multiple tasks and will contain guided project pictures to explain my work.

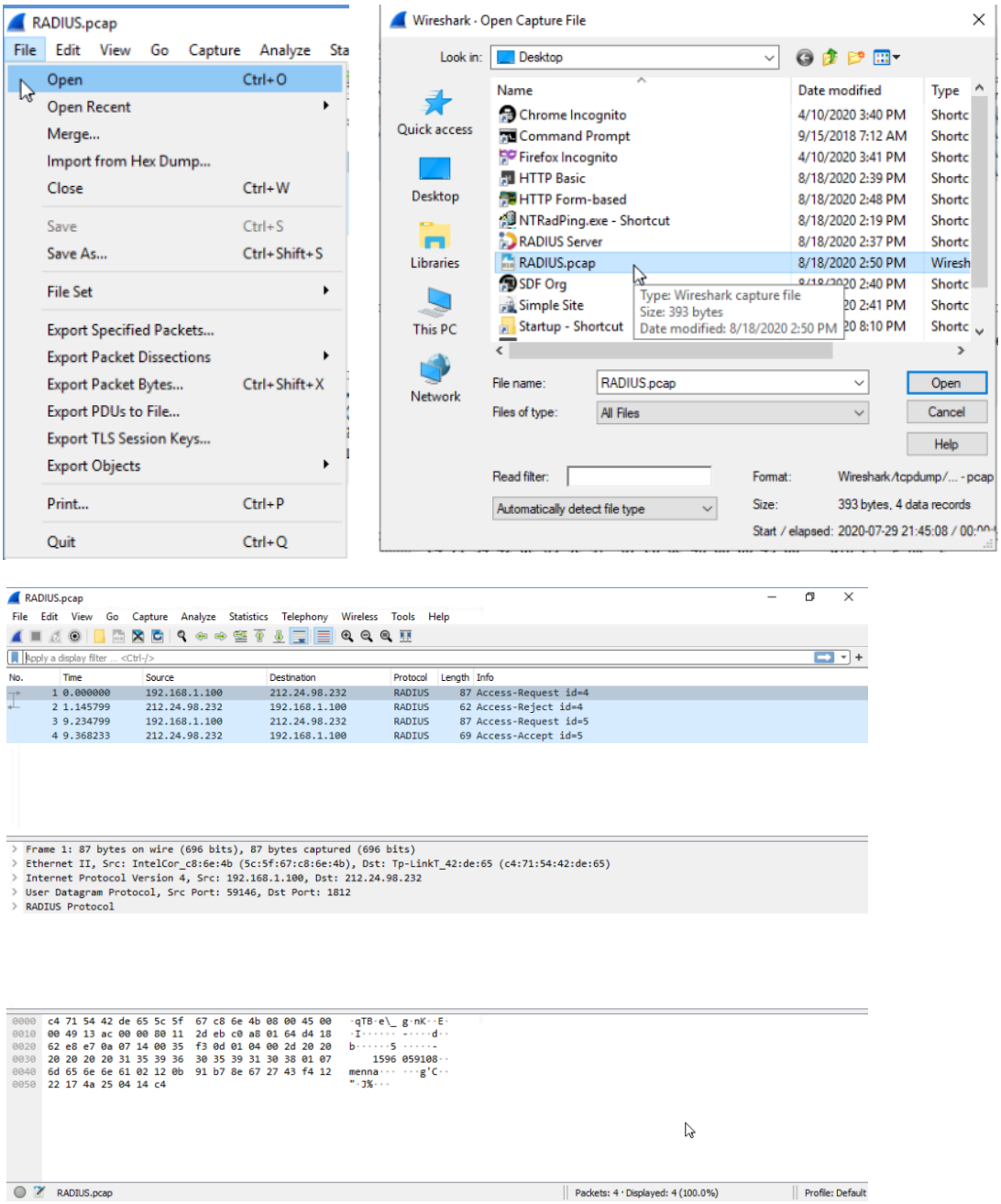
Tasks/Objectives:

- Get to know Wireshark and its basic functions
- Generate and Capture RADIUS Traffic
- Analyze a HTTP Basic Authentication
- HTTP Form-Based Authentication and DNS
- Initiate, Capture and Analyze Telnet Sessions
- Capturing and Analyzing SSH Sessions
- Generate, Capture, analyze ten decrypt HTTPS Traffic

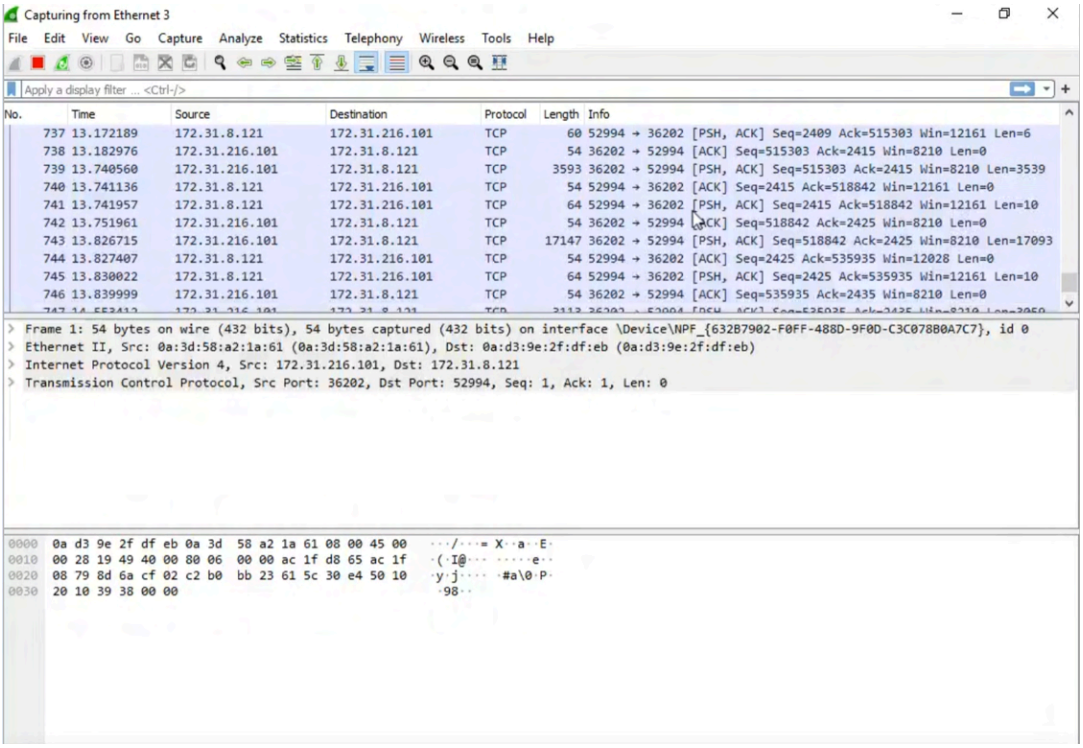
Task 1 - Get to know Wireshark and its basic functions



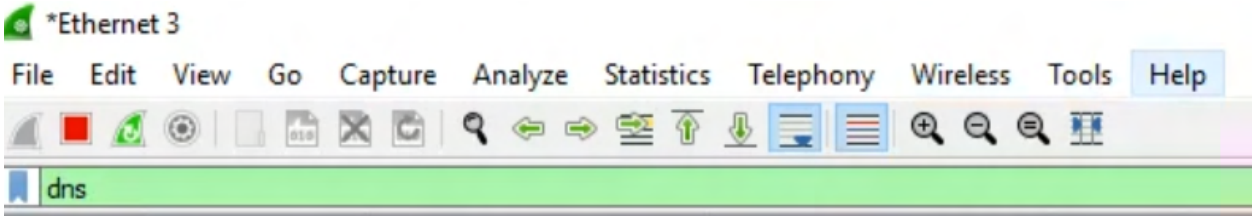
In this project, we will be using a Cloud workspace which is installed with a version of Windows 10. In here there are a couple programs already installed such as Wireshark, FireFox, Google Chrome, RADIUS server and a few others.



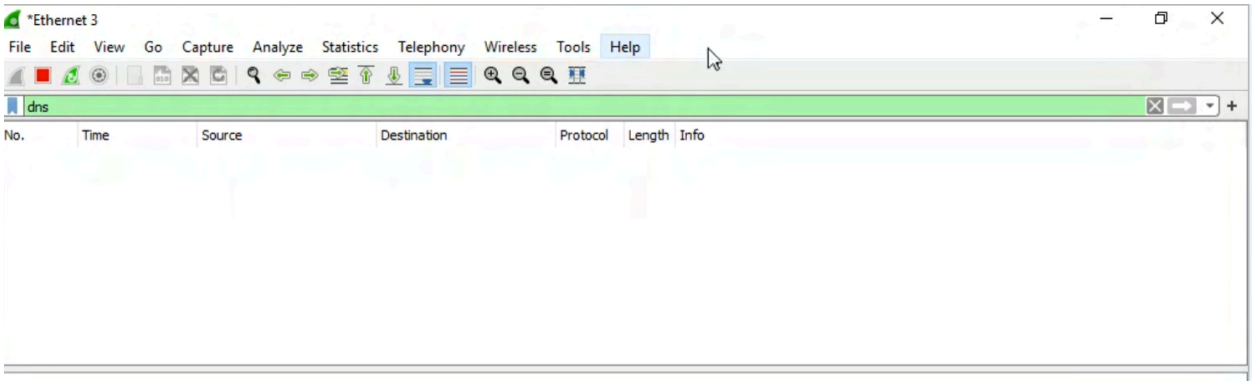
In the images above we are prompted to open a `.pcap` file from `RADIUS` on our desktop. To open this file on WireShark we must first go into the file button on the top left hand corner. Once in “file” we should be able to navigate to open. Once the open application is available we must go into “desktop” and click on the `RADIUS.pcap` file which will open our desired file onto Wireshark.



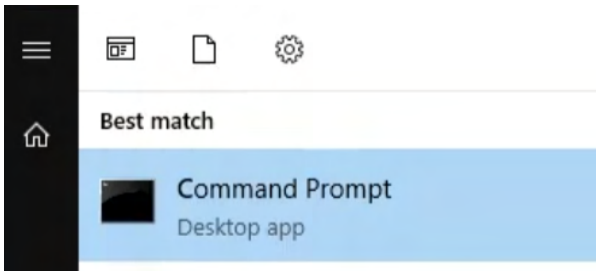
Next, we are prompted to open a live capture on Ethernet 3 network. We can double click on Ethernet 3 on the home page and it will bring up a live capture of internet traffic that is being captured at the moment.

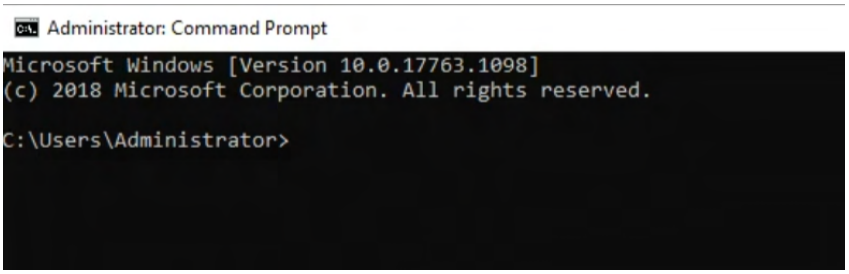


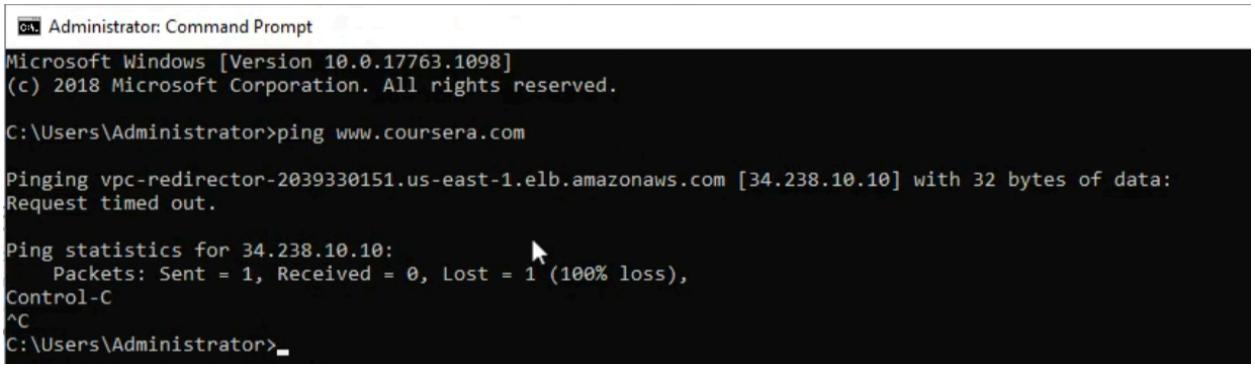
To continue, we are asked to see if we are able to use the display filter box. For this exercise we are prompted to use “dns” as an input into the display filter. (As shown in the image above) Funnily enough, when inputting dns, there is no live capture happening which is why there wireshark is left blank. (See image below)



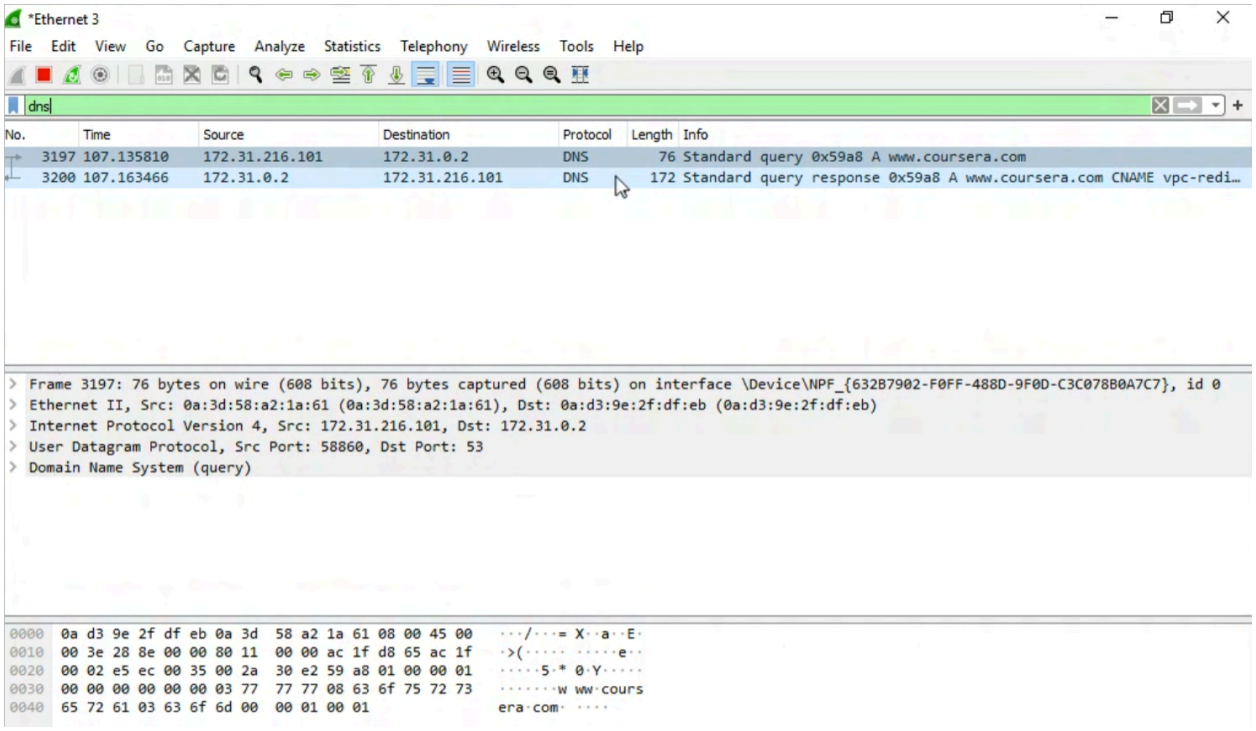
We can try to bring in some captures by pinging (Coursera.com)
First you must open cmd prompt





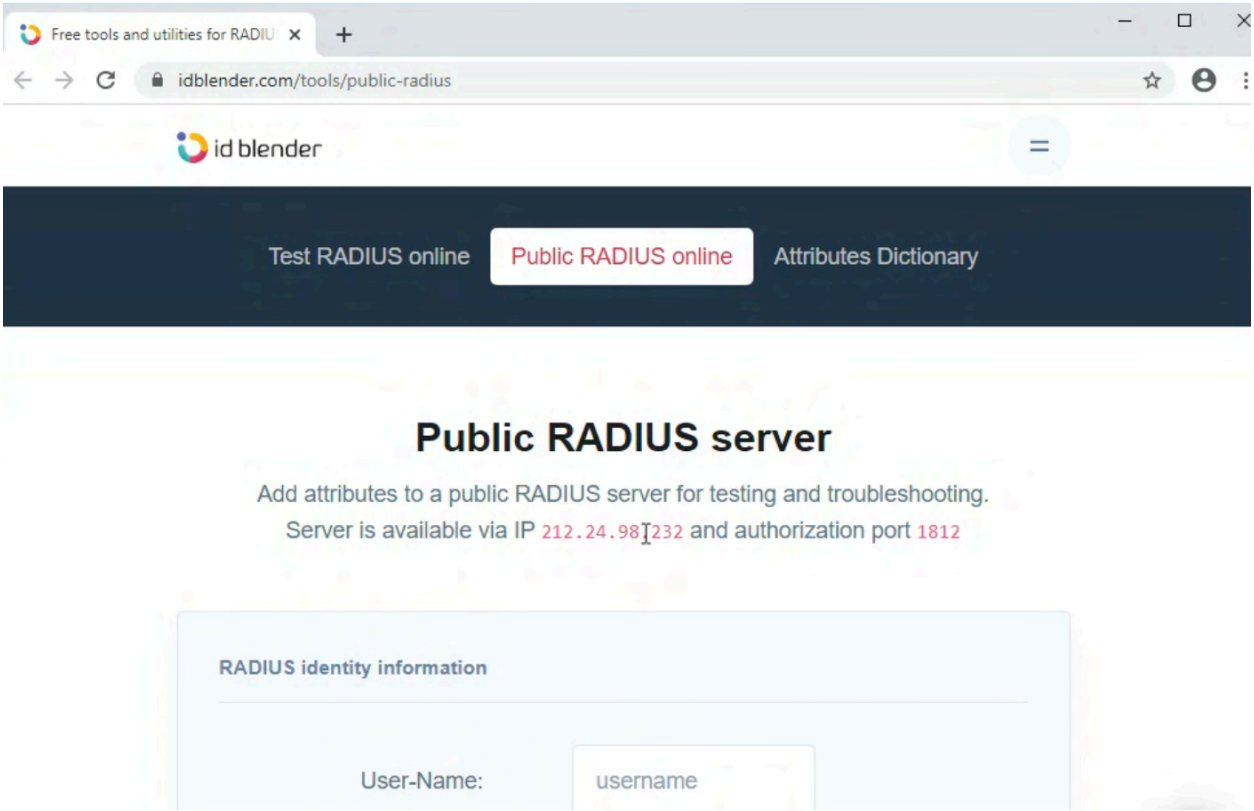


After inputting the command `ping www.coursera.com` we are able to see two new captures on Wireshark. (Image Below)

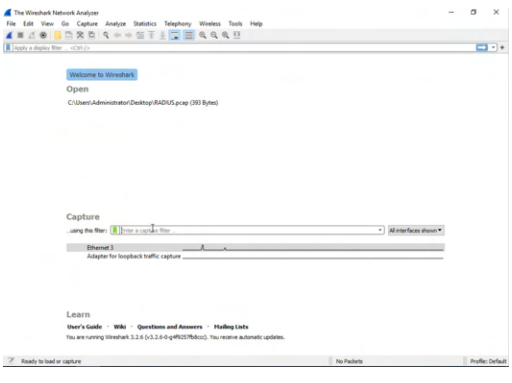


With this, we can conclude our first task, which is getting to know Wireshark and its basic features.

Task 2 - Generate and Capture RADIUS Traffic

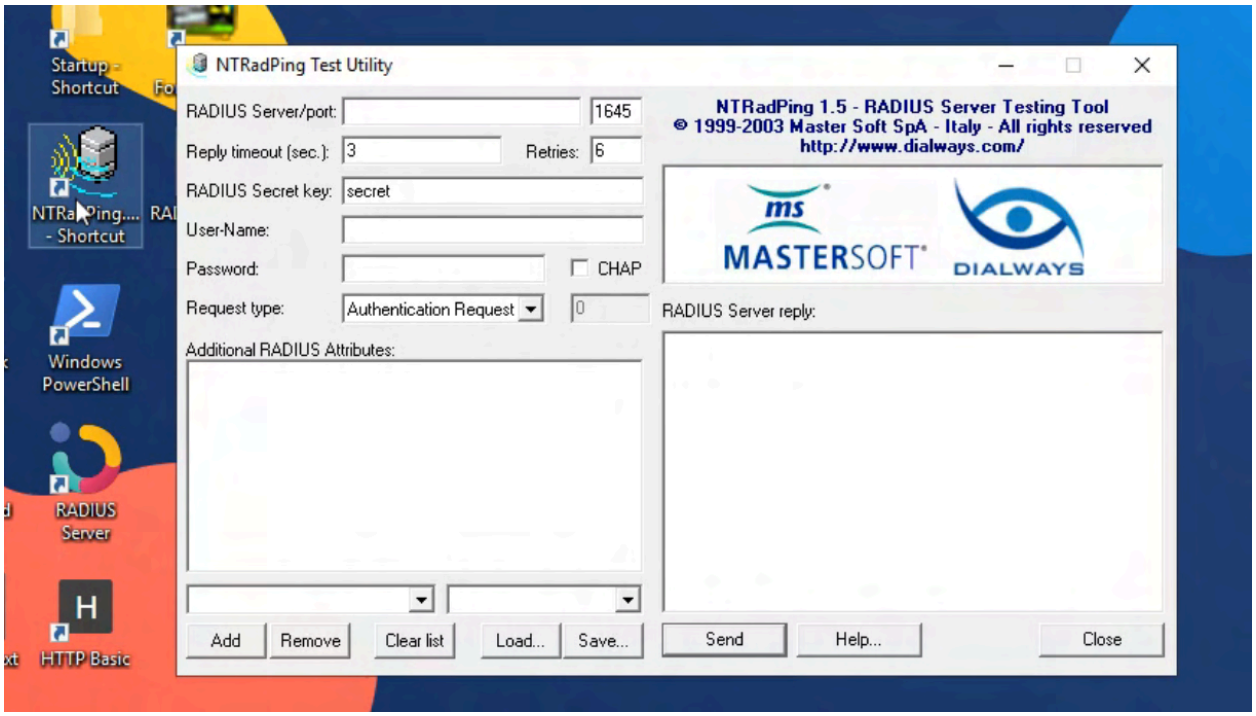


First we are going to open the RADIUS server. Doing so will open up a browser with RADIUS GUI Home page. We can note that the Server operates on the IP of 212.24.98.232 and Port of 1812.
At the Bottom you are able to create your RADIUS server account - For this project, I chose to Use a generic username and password.
Before clicking the Submit button, we should first head back into Wireshark.

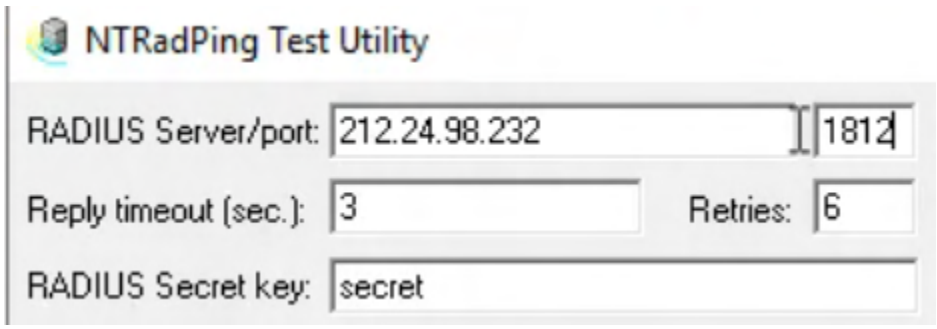


While in Wireshark we should be able to capture ONLY the RADIUS network traffic. How can we do this? We would need to use a filter before capturing network traffic. As stated previously we can see that the RADIUS server operates on port 1812.
We can use this information to input the filter `port == 1812` or `port 1812`

However, there is a second step you must follow before capturing RADIUS traffic. We must open the NTRadPing Shortcut/Application which would look like this:

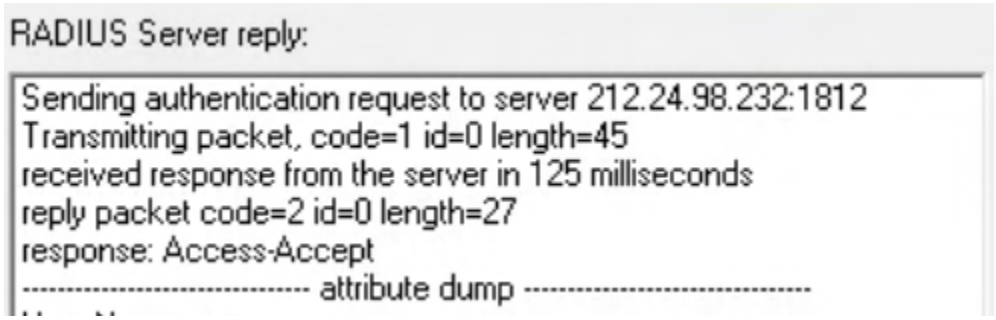


This is what we will use to send out an authentication request to the public server we have created the user on.

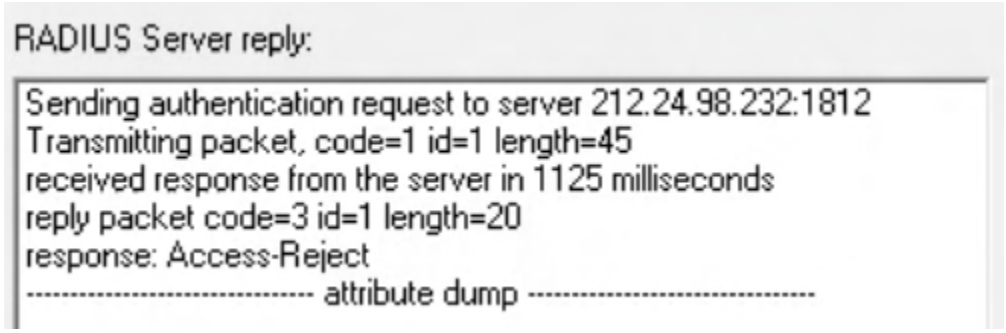


Here at the top we have to input the RADIUS Server and the port number. We will leave the reply timeout and retries to default (3) and (6)
We will also leave RADIUS secret key to secret

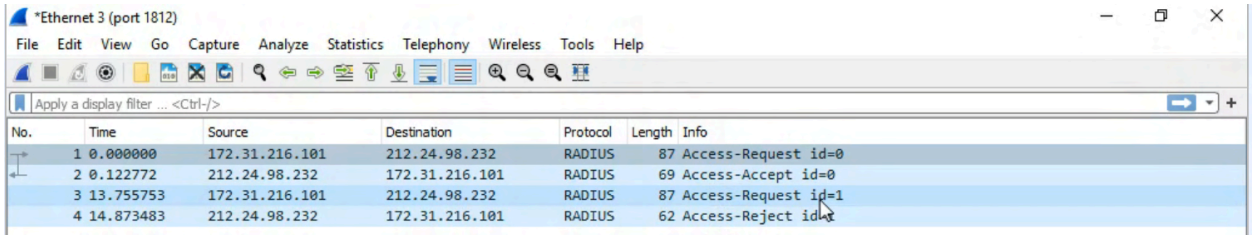
Moving forward - you may input the Username and Password that was created within the RADIUS home page GUI. For the project, i have chose a generic username and password. For example - you could use “admin” for the username and “password” for the password.
On the right hand side - we are able to see that the response for access has been accepted

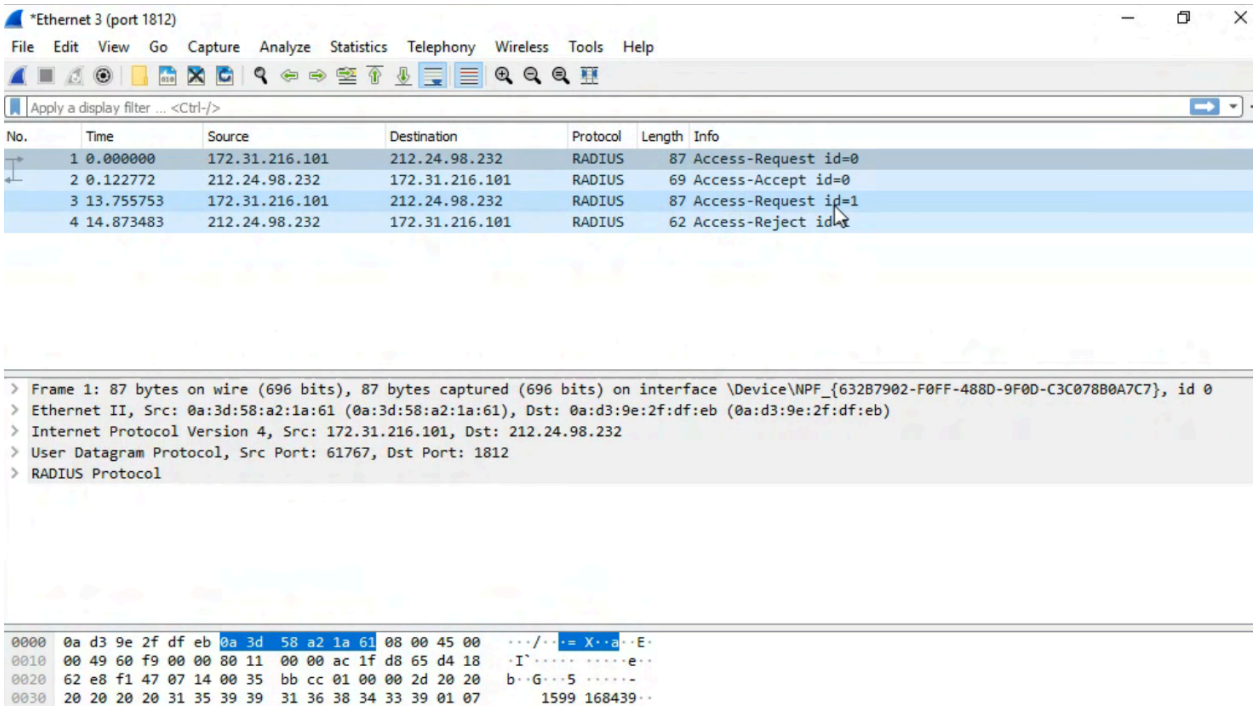


If we were to put the incorrect password we will get an access rejected response shown here:



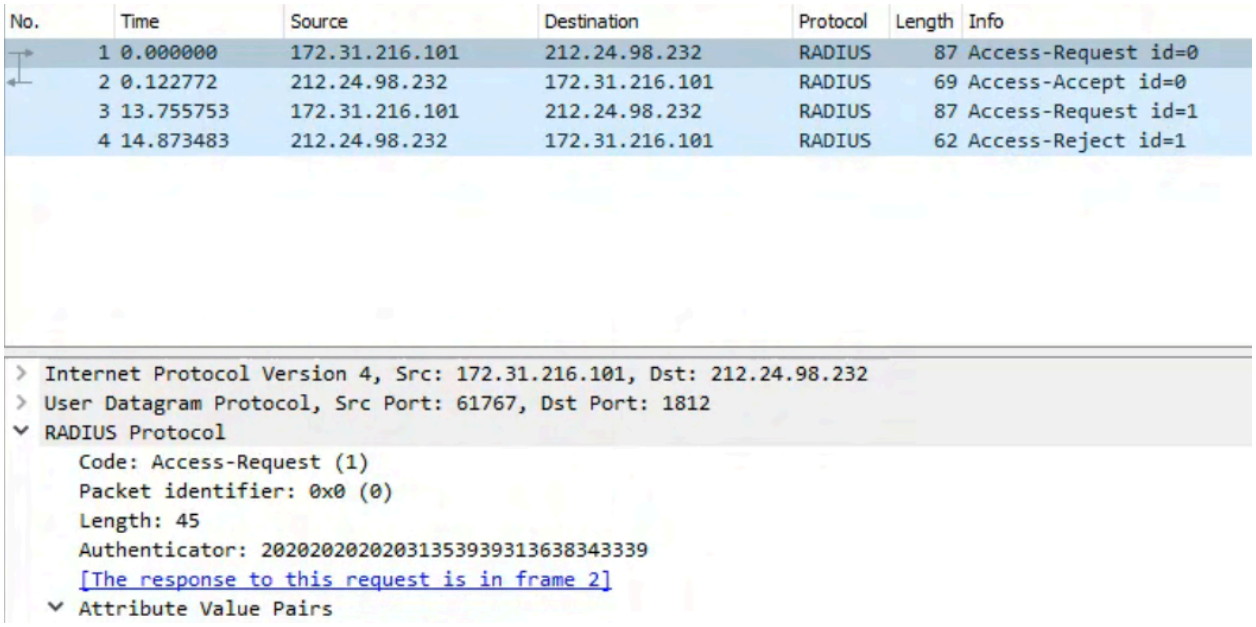
With Wireshark running and activating RADIUS server ping we are now able to see entries within our network capture which looks like this:





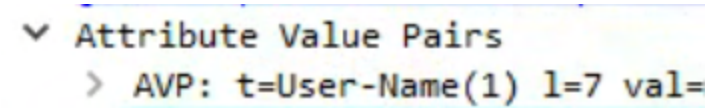
Notice how in the entries we are able to see two Access Requests. The first of which, which is highlighted, was accepted and the second request was rejected.

To dig even deeper, we can also view the authentication request username in the following field

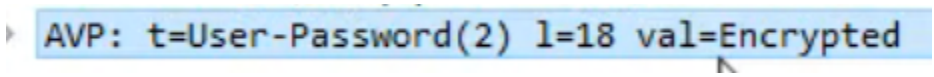


Underneath the final entry, it would display the username we used to authenticate an access ping

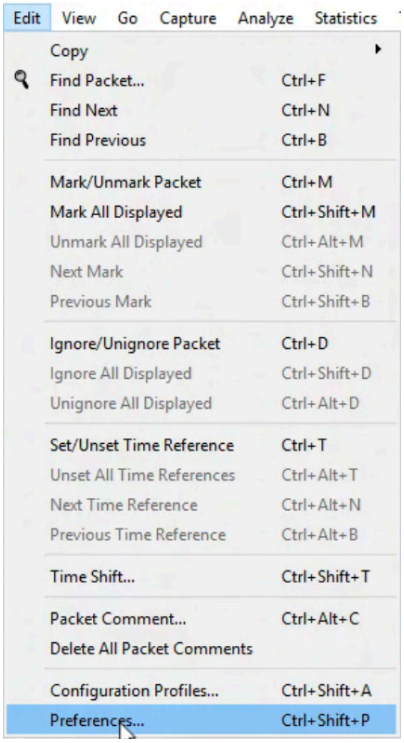
For this demonstration I will not show the username used.

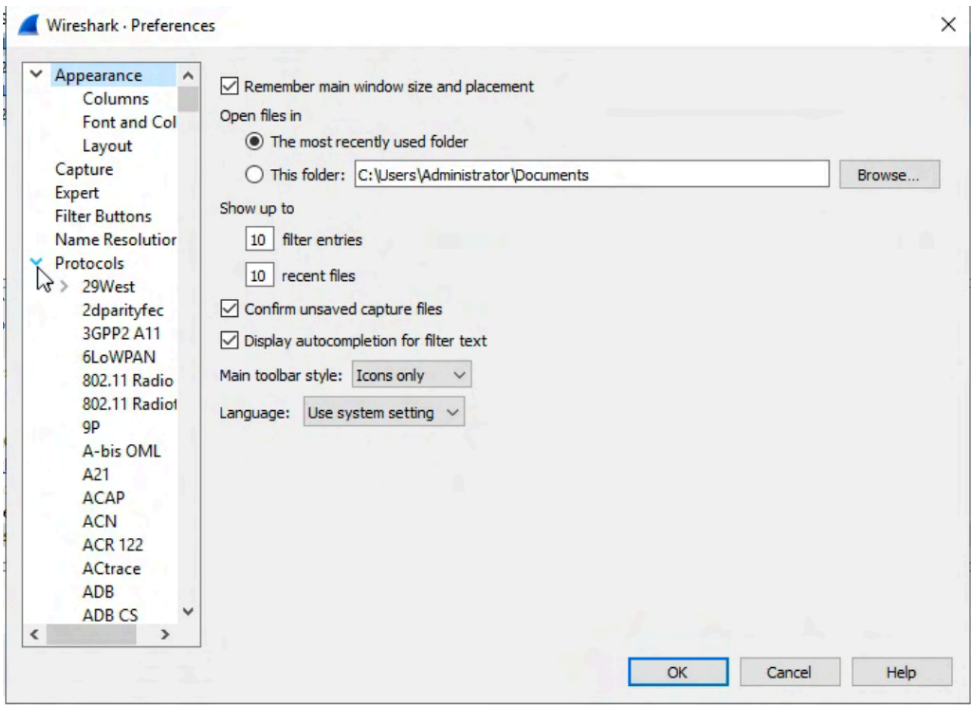


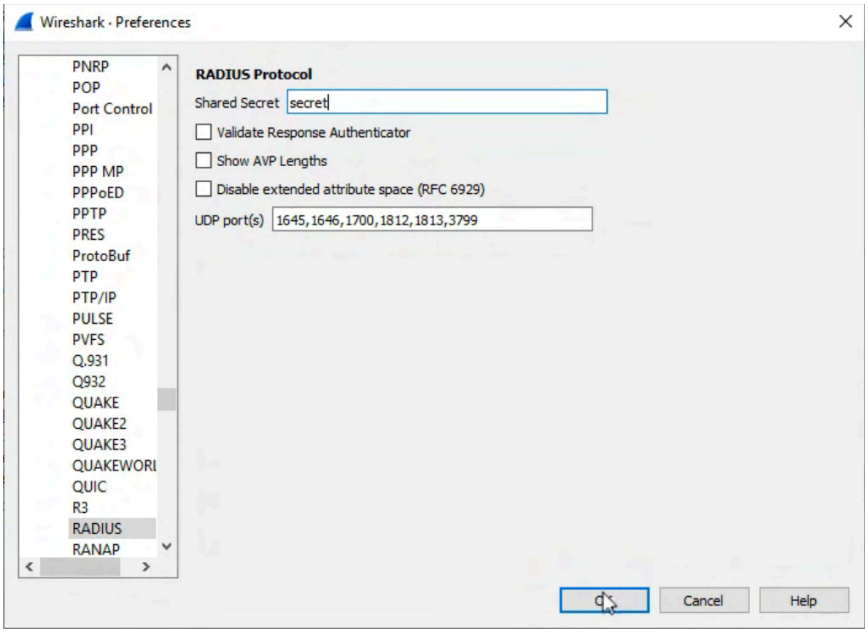
Also along with this, there is a entry for password used however it is displayed as encrypted.



However, there is a way to display the password that was used. To do so: go into the edit button -> preferences -> protocols -> search for “radius”
Within shared secret, we will input the shared secret password. For this demonstration it was “secret”. It will display the decrypted password after following these steps. Here is the photo demonstration:







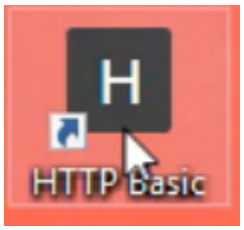
Task 3 - Analyze a HTTP Basic Authentication

In this task, I am going to demonstrate Wireshark's capability to analyze HTTP traffic and dive into Basic Authentication

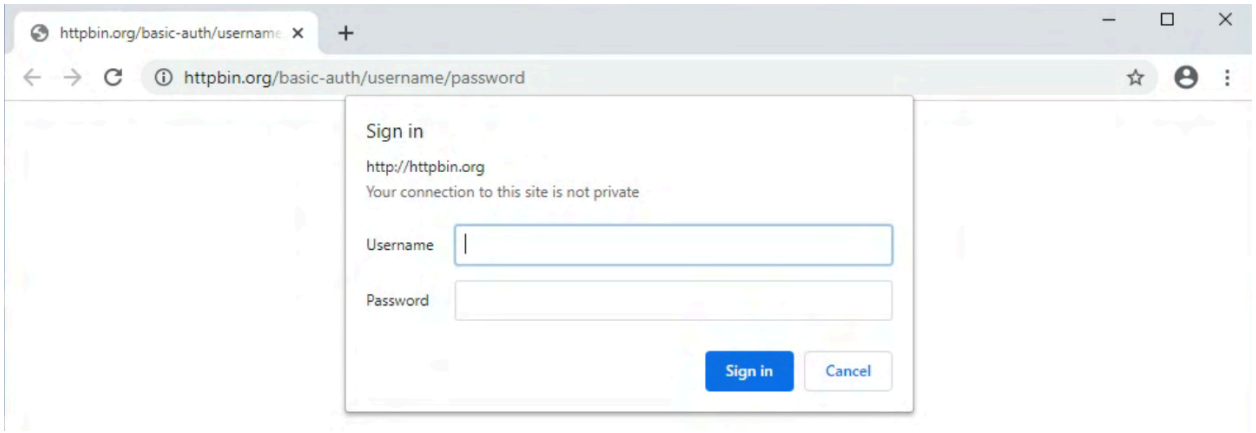
As we are informed, HTTP traffic is unencrypted. We may notice how sometimes when using a web browser, Microsoft's Windows Defender will warn their users when clicking into an HTTP link which may be insecure. For this task, we will be following a couple of steps to identify authentication traffic between a HOST to a HTTP server/website.

[Good things to know: HTTP = Port 80 - HTTPS = Port 443]

To begin, we can head back to our project workspace and click on the HTTP Basic Icon:



When opening this application, it will bring us to the website on our browser:



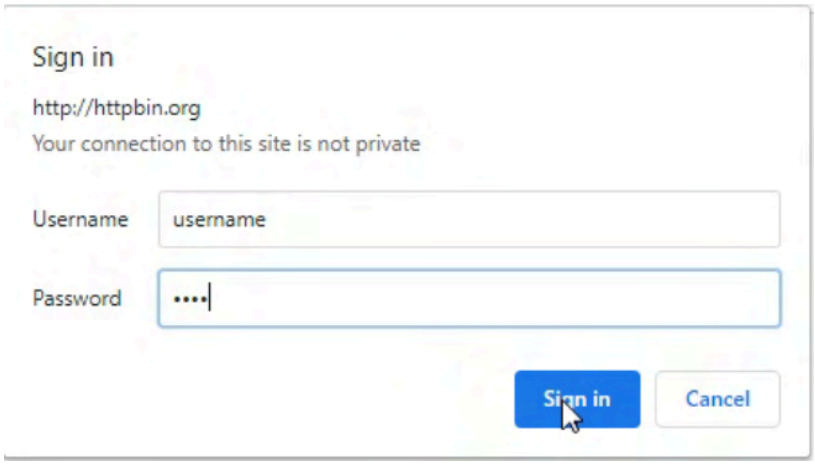
Note that for this demonstration our username and password is "username" and "password" (Very insecure **DO NOT DO THIS FOR ANY PERSONAL LOGINS**)

Before entering our credentials, we should start out Wireshark Capture.

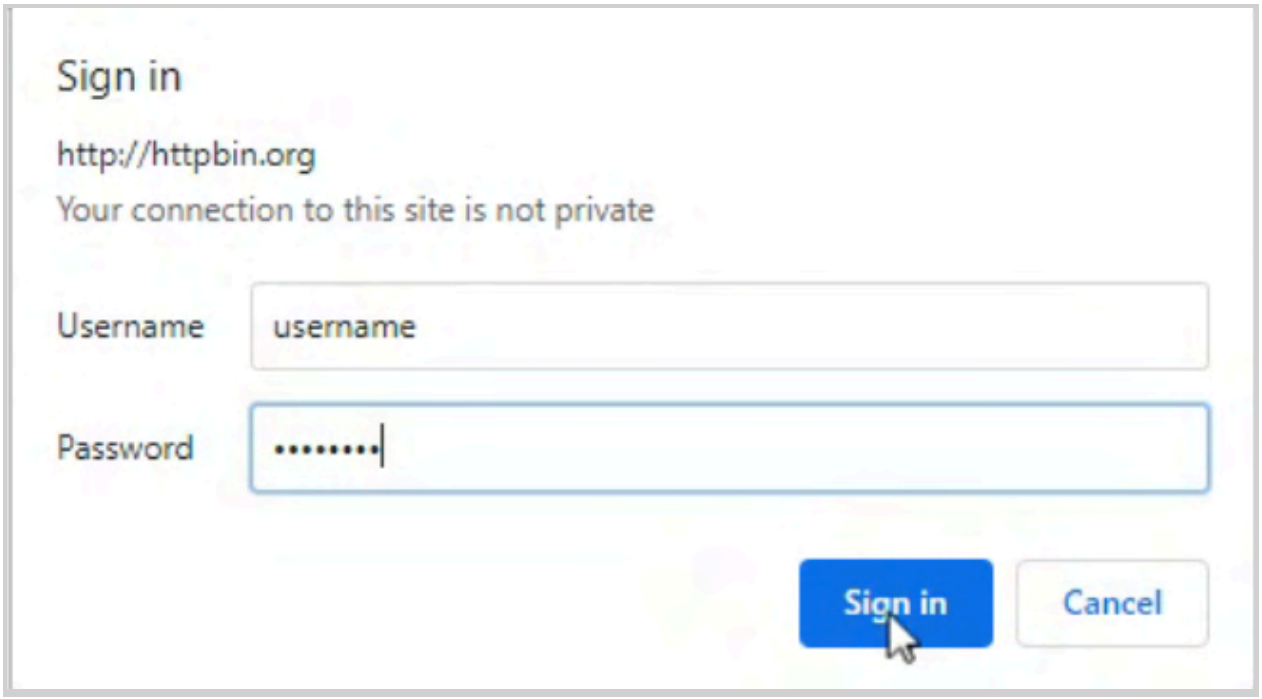
Since we are looking for only HTTP traffic, we can apply a filter before capturing.

We may use either: `port == 80`, `port = 80`, or `port 80`. All will do the same exact thing.

Our first demonstration will begin with the us entering the incorrect password to our login credentials.

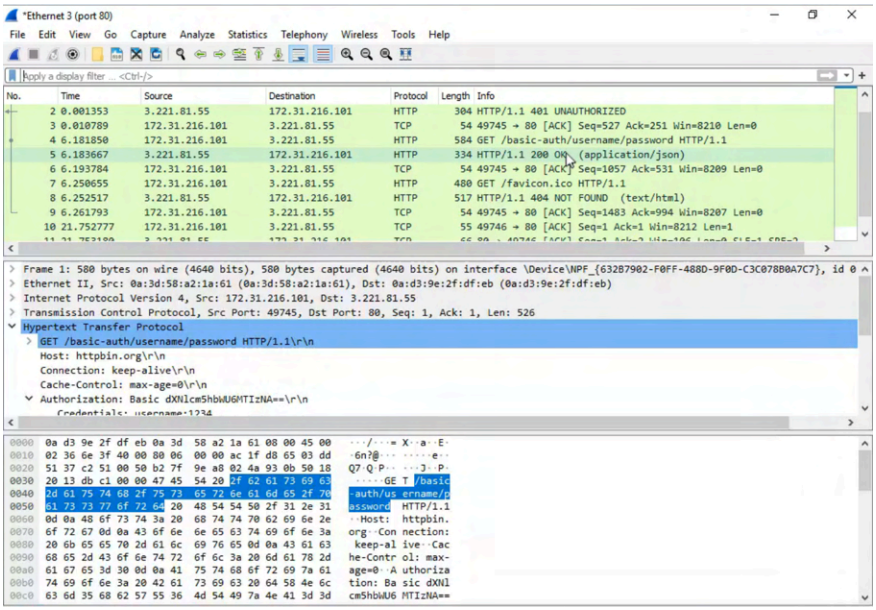


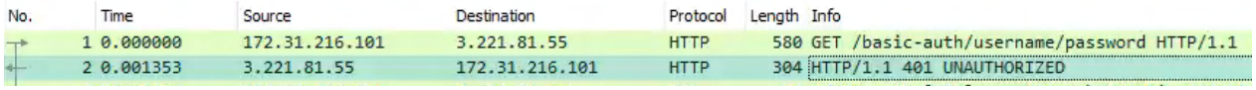
Here, we have used the correct username however, we used the wrong password. We have inputted: `1234`



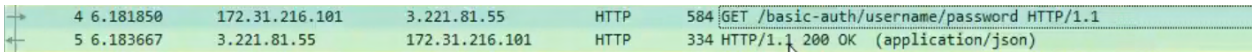
When entering the correct username and password, we are granted access to the webpage. Here in the final photo we can see that the authentication is displayed in the home page.

Lets head back to Wireshark.
As you can see on the right hand side, we have stopped the capture and is met with multiple captures. Please notice how all the traffic is unencrypted. All credentials can be shown and everything we do can be shown and read.





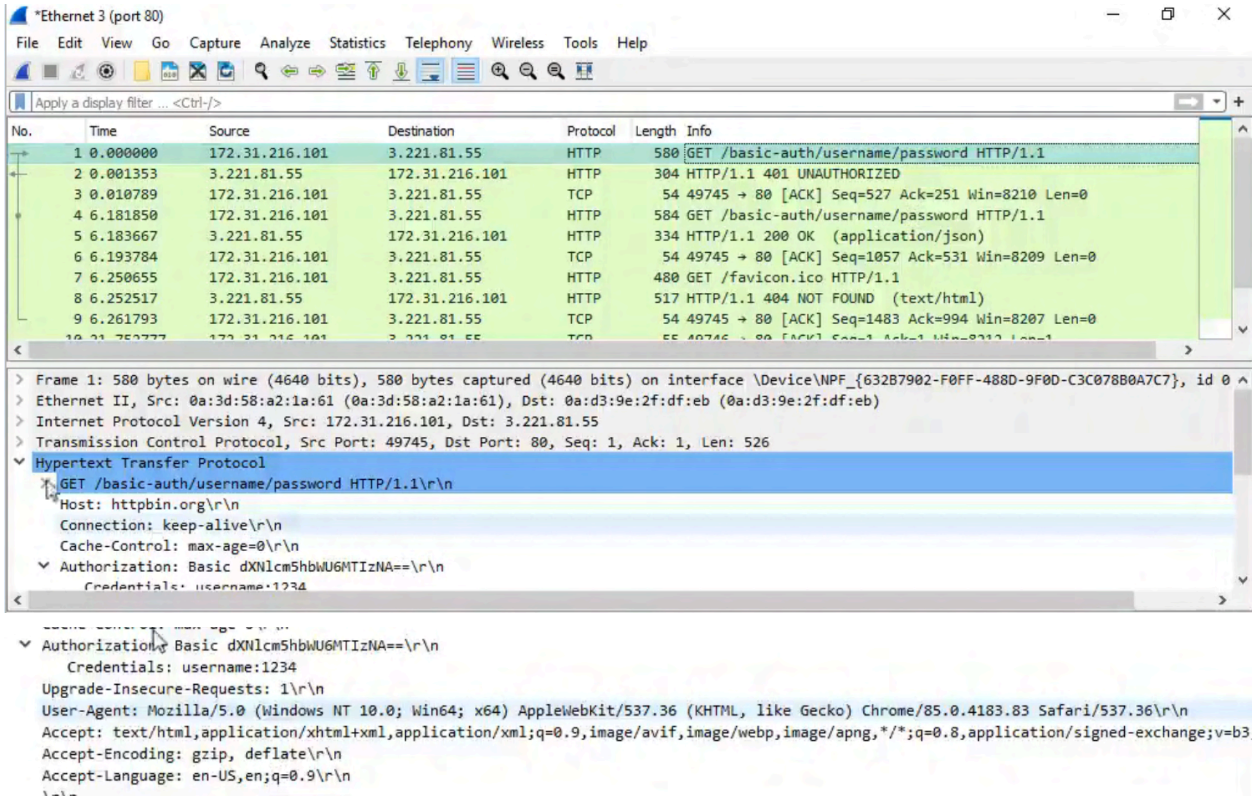
With our first entry we can see that we have sent out a authentication request and it was denied due to our failed attempt using the wrong password.



However, on our second attempt we can see that the authentication was accepted. Which is displayed with the `HTTP/1.1 200 OK`

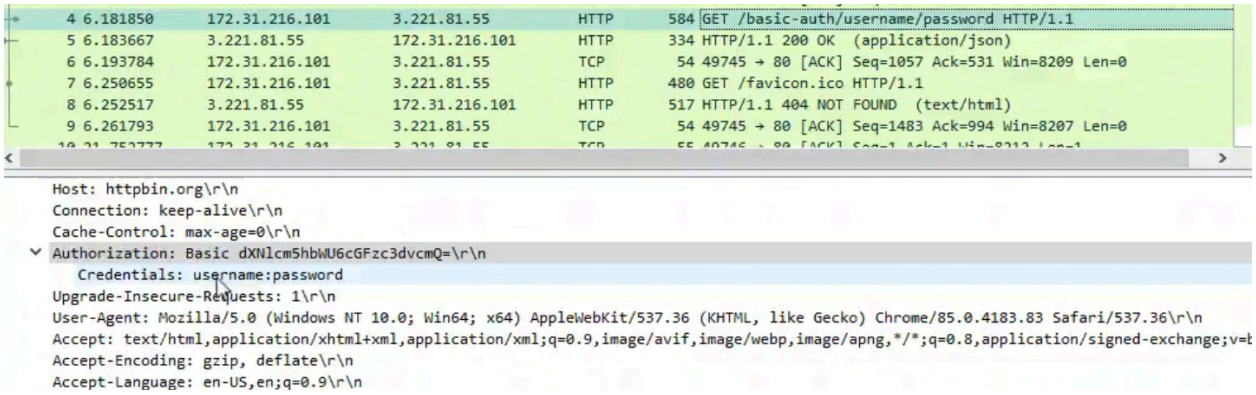
This is very insecure. If we were to use this same procedure on an HTTPS server we would not have been able to see what has happened.

To further emphasize how insecure this is, we can click back onto the first entry and expand the Hypertext Transfer Protocol and the Authorization tab.



When doing so you can see the credentials and the password that was used for the first attempt.
Right next Authorization we can see the Basic encoding, which is not encrypted, of the username and password that is being sent to the server.

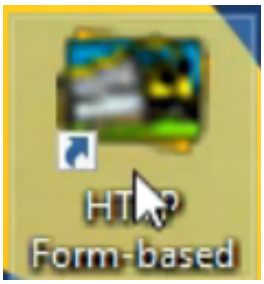
When clicking onto the entry with the correct credentials:

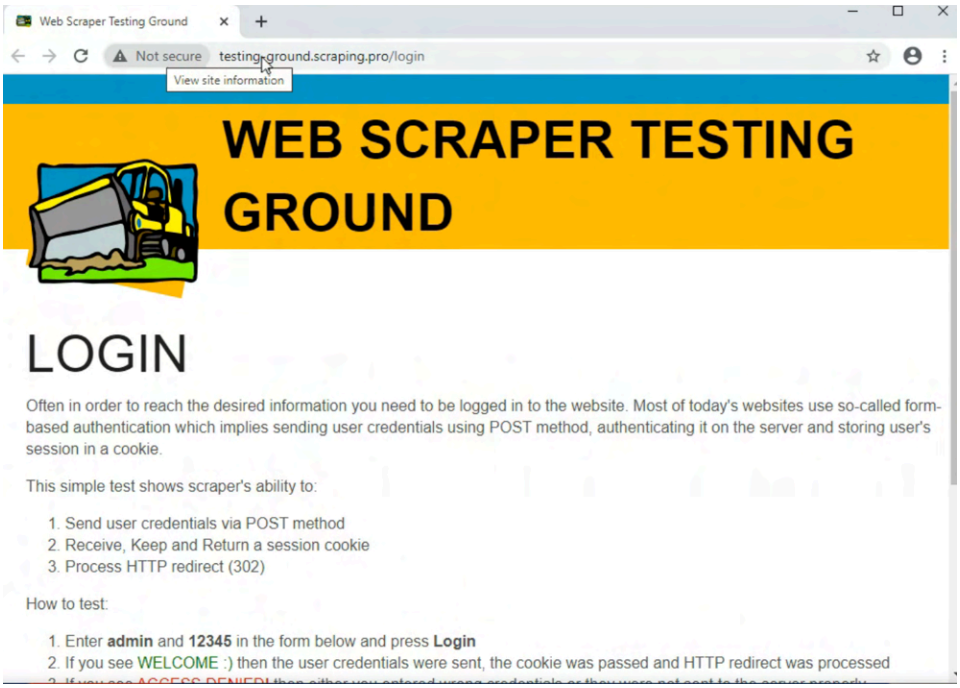


We can see that correct username and password, along with the basic encoding that was sent to the server

As a threat actor, we can assume that HTTP is very easy to swipe someone's credentials and create a very dangerous situation to both personal/citizen or to an organization

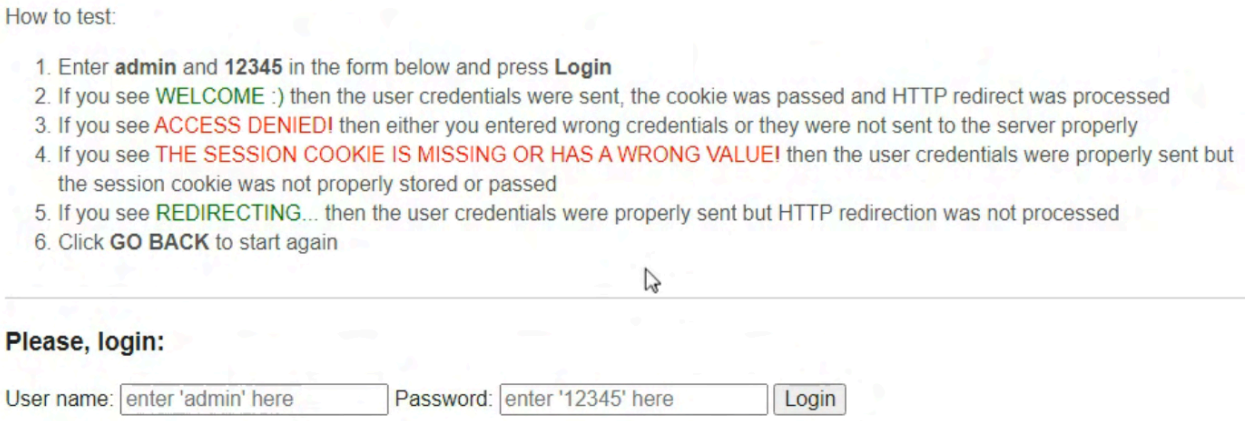
Task 4 - HTTP Form-Based Authentication and DNS



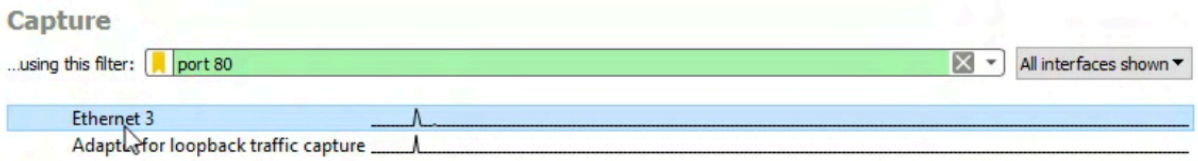


Form based authentication which was created to mitigate the downsides of basic authentication. Form based authentication uses the standard HTML form fields to pass the username and password values to the server. This does not send a get request with the credentials, instead it will send a post request and will pass the request in the post body.

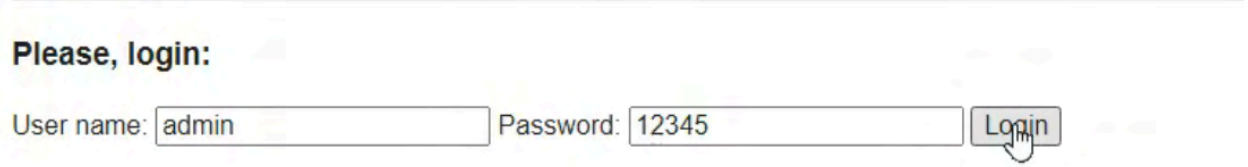
In the images above, we are opening the HTTP Form-based shortcut which will open up a browser and a HTTP website. We can notice that the website is not secure as it uses HTTP. When scrolling down the page we can see instructions on how we can test this website and a credentials field. Shown here:



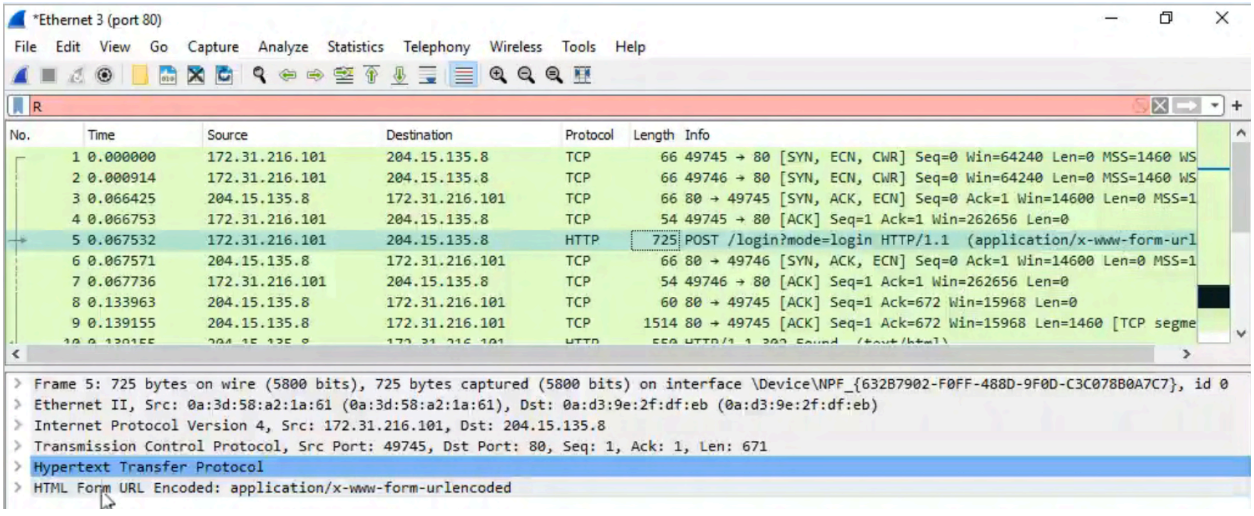
Before inputting the credentials, we should begin the Wireshark capture. Using the filters before capture we can capture only HTTP traffic using the query we used before. `Port == 80`, `Port = 80`, `Port 80`



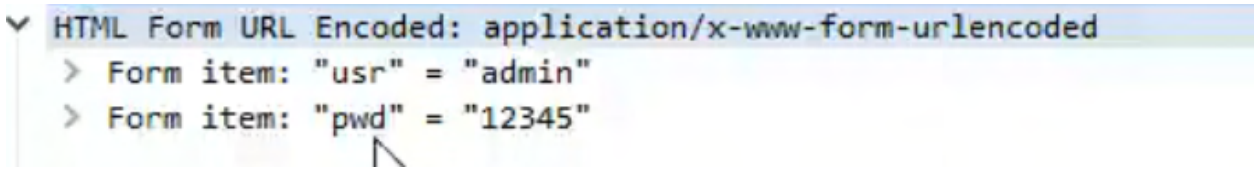
After enabling the capture, we can go back to our test website.
We can now input the credentials: Username = `admin` and Password = `12345`



Once we have finished entering our credentials and clicking on the “login” button we can stop Wireshark capture.
In our previous task, we can note that we were able to find the credentials within the basic authentication inside the hypertext transfer protocol. However, this time we are able to find this using the HTML Form Fields.

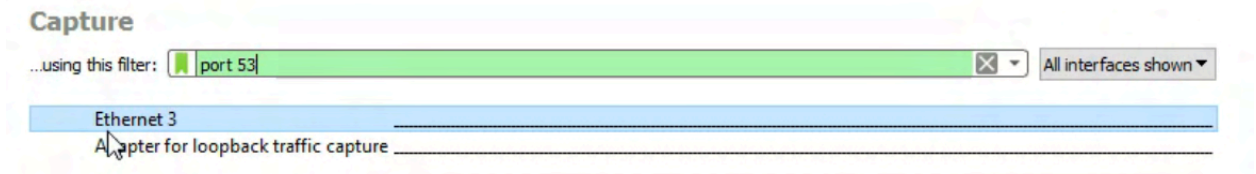


In the image above, it is shown that we have clicked on the HTTP protocol, length of 725. This is the capture entry of our authentication request. We can see that in the bottom there is the “Hypertext transfer Protocol Drop” down menu as well as the “HTML Form URL Encoded” drop down.
When expanding the HTML Form URL Encoded, we can see the form item of both the username and password:

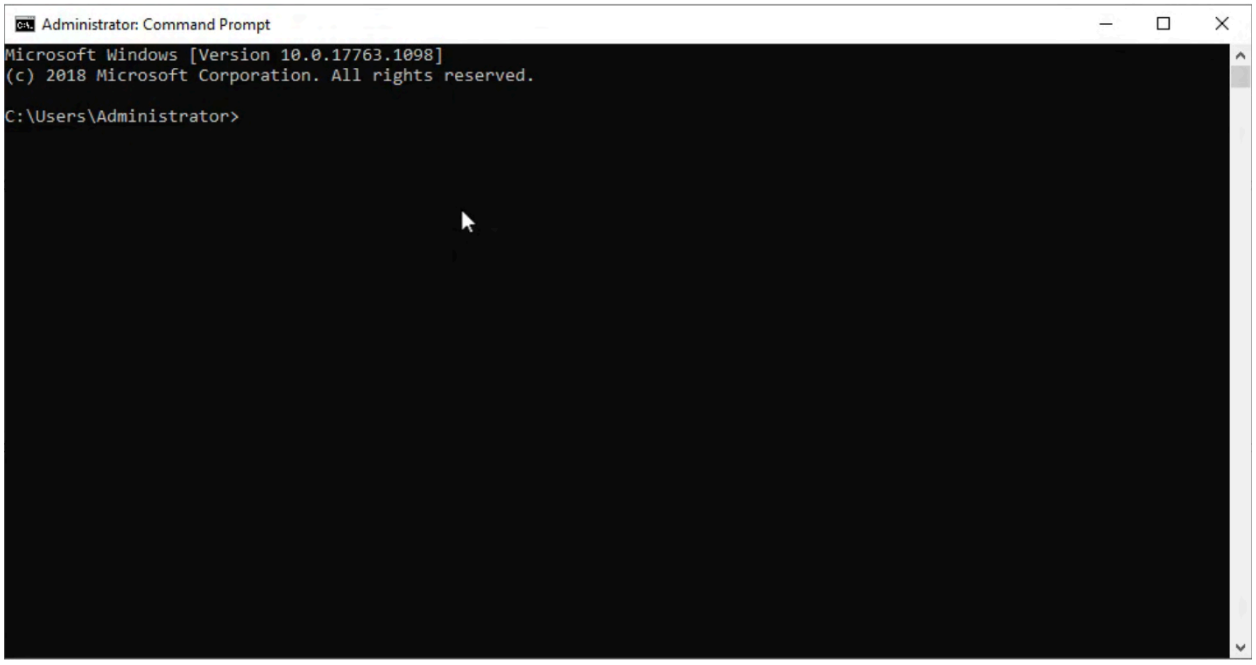


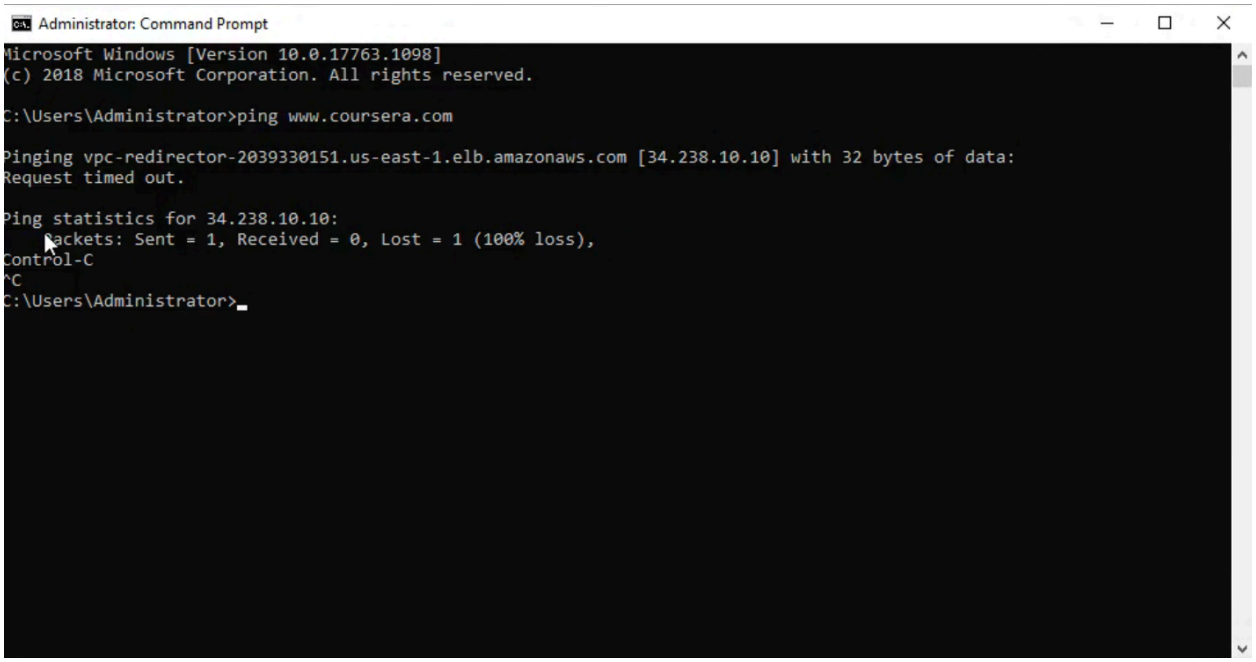
While this is procedure is even less secure than HTTP, HTML Form was developed with tuse case of HTTPS in mind. This is because the entire packet would already be encrypted anyway so there was no need for worrying about any encoding.
This is a great reminder to never input important/sensitive information online that is using HTTP There are probably 3rd party games, pirating websites, even pirated streaming platforms that would require user credentials which can be insecure due the nature of HTTP.

To continue this task, we can begin DNS traffic capture.

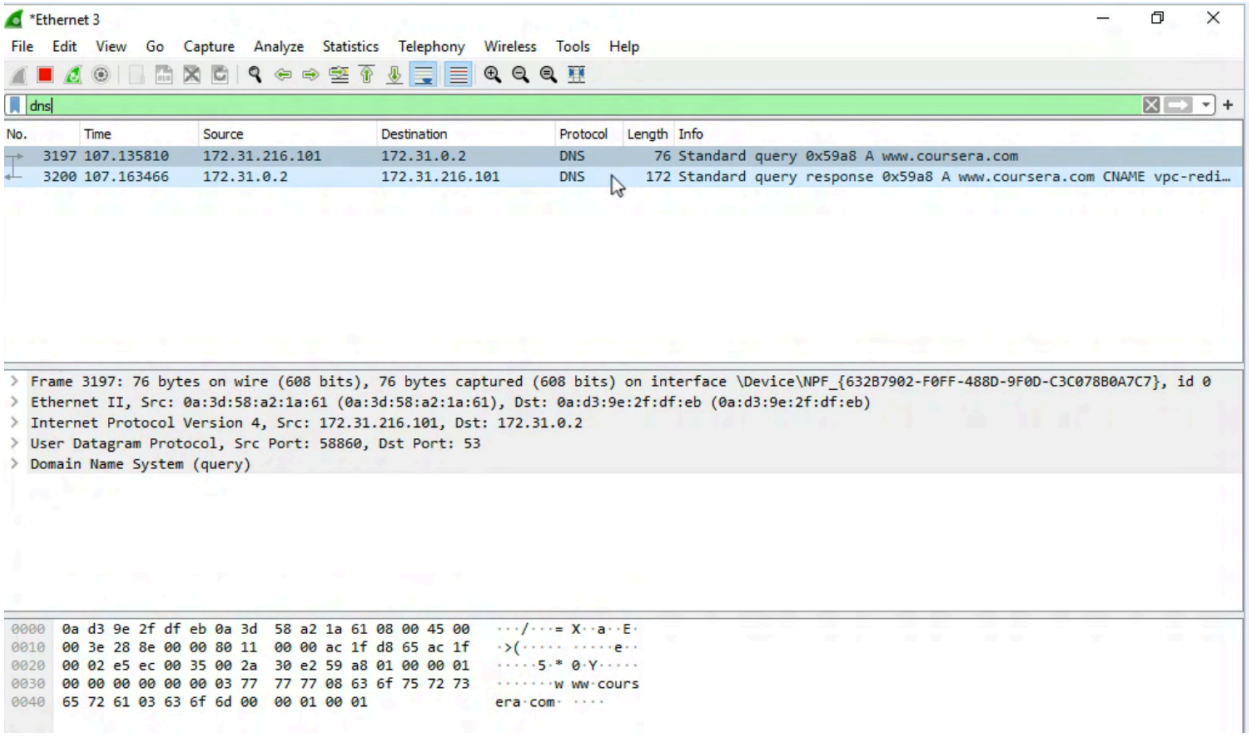


Notice how we are now jumping to `port 53` for DNS traffic.
To begin, we should open the Command Prompt within our desktop.





In here we can input the command `ping www.coursera.com`
This will allow us to capture DNS Traffic on Wireshark



As shown above, we have inputted the display filter `dns` onto our Wireshark application which should only display our DNS ping.

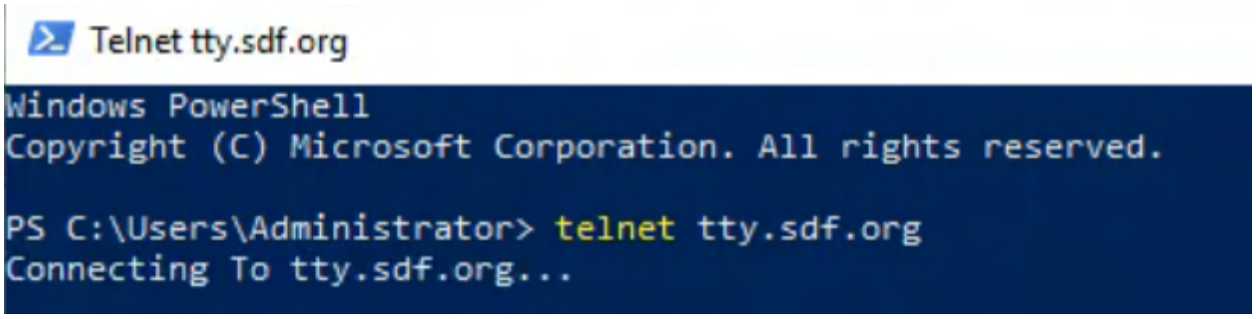
Task 5 - Initiate, Capture and Analyze Telnet Sessions

To further our understanding on unencrypted protocols, we will proceed with our 5th task which will revolve around Telnet. We will be experimenting with Telnet as well as comparing it with SSH in our next assignment. Telnet is a computer protocol that was built for interacting with remote devices to access and manage said device. In this use case, we will be using telnet to `tty.sdf.org`

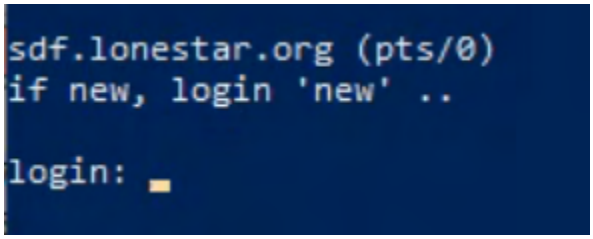
To begin we can open the shortcut on our desktop SDF Org

We are presented with the credential login page for “Free UNIX shell Account”
Telnet operates on TCP Port 23

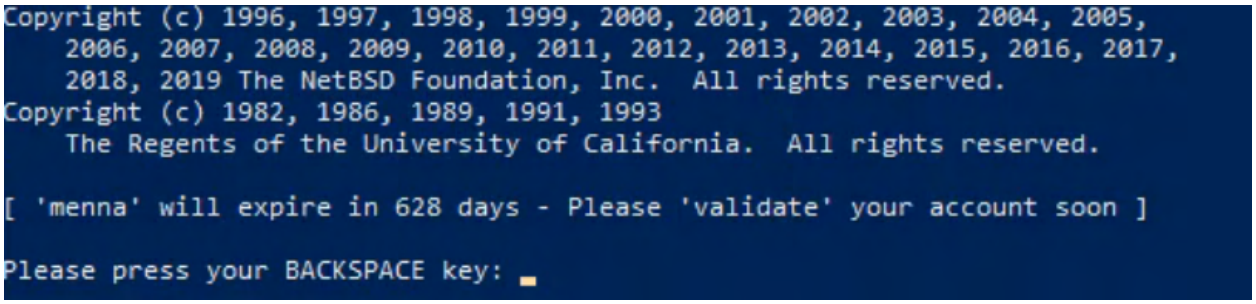
To continue capture telnet packets we will be using windows Powershell



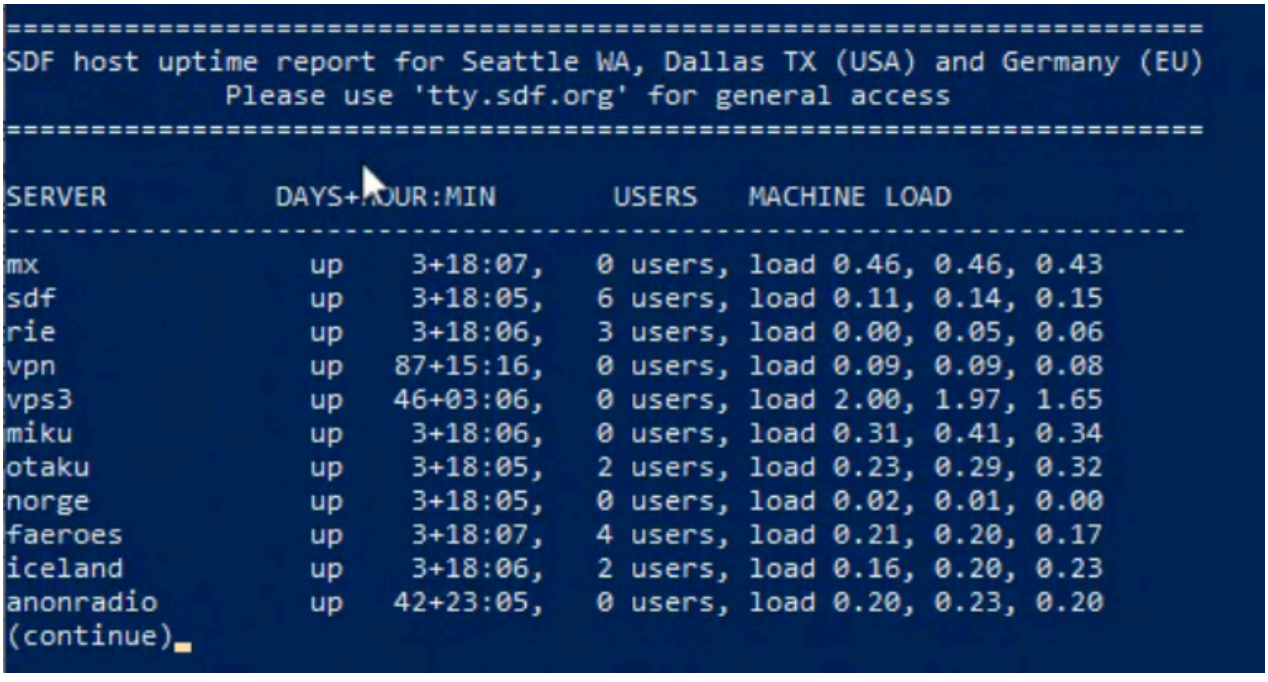
When using, telnet in powershell we can also place the website url to the command which will connect our desktop with the corresponding URL.
You will then be greeted with:

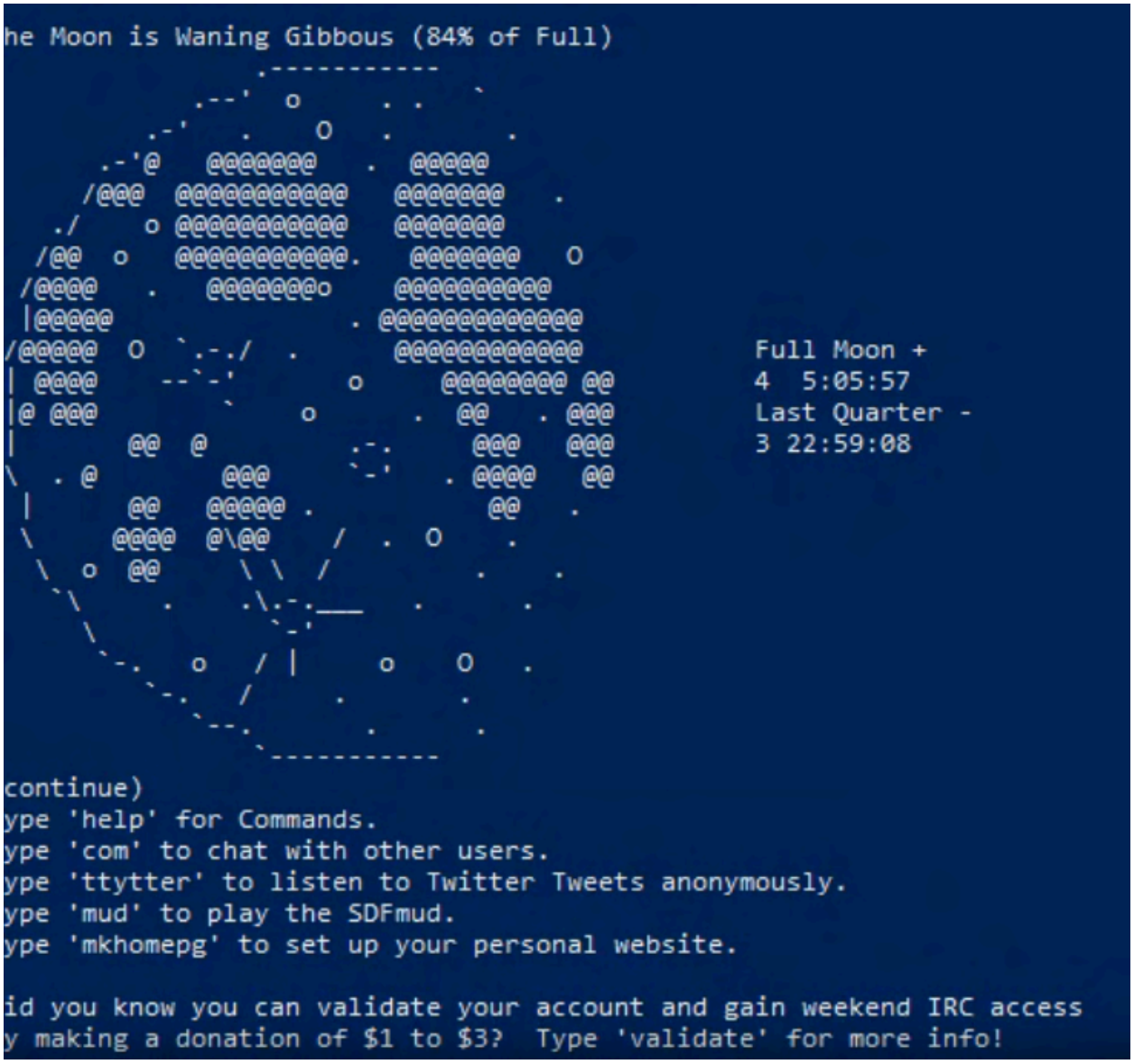


You will enter the Username and password for this session
Make sure that the password is case sensitive
When inputting the password there will not be a display of key inputs

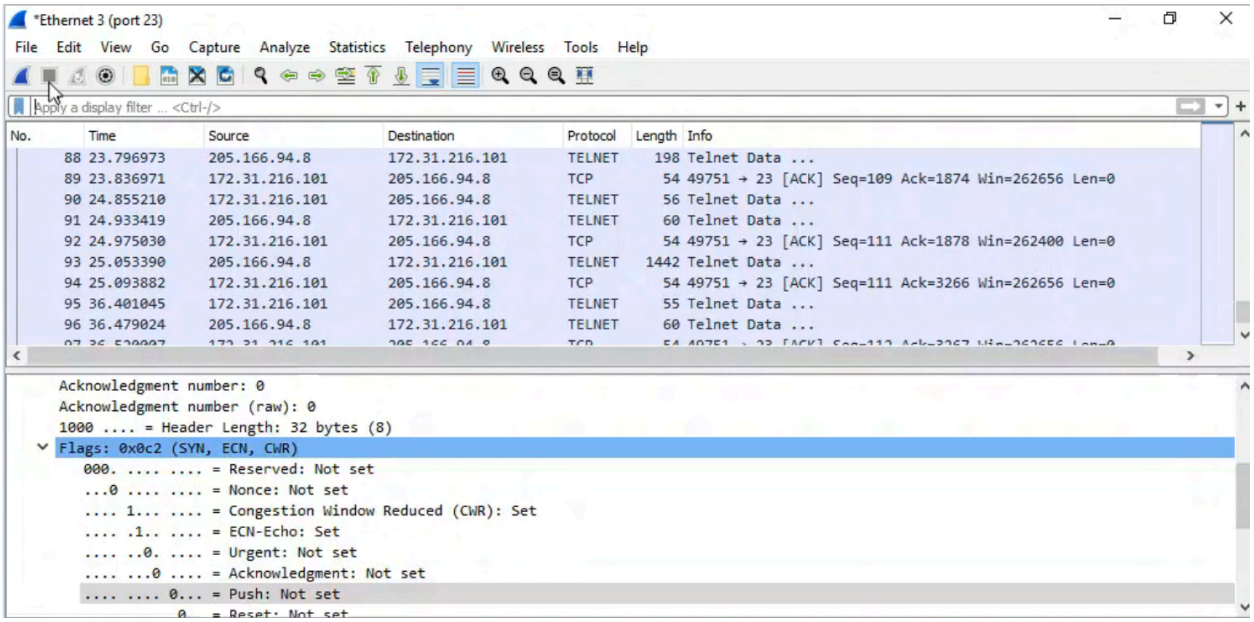


After inputting the correct credentials there will be a prompt to enter the backspace key





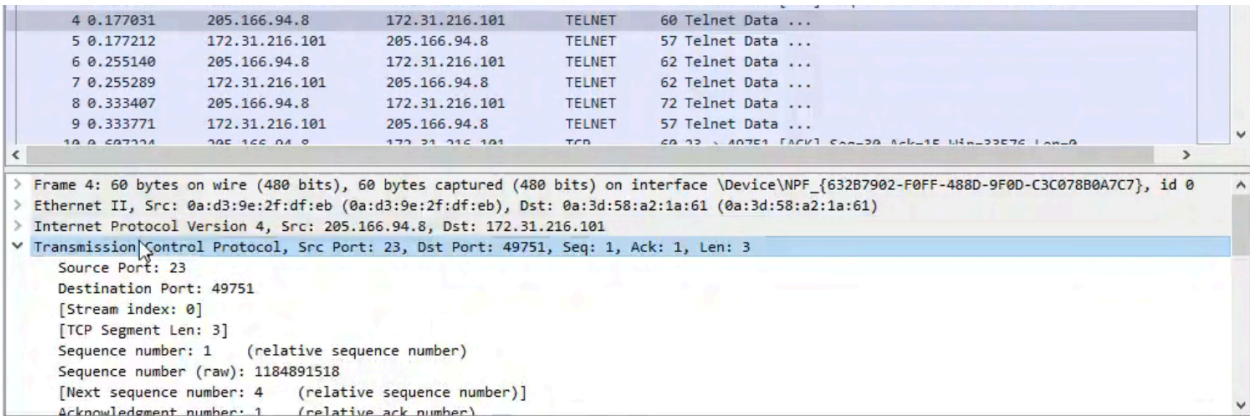
With the images above, I have followed the prompt commands to continue with the telnet command. In the final image we are able to see a graphic of the moon. This will conclude our Wireshark capture.

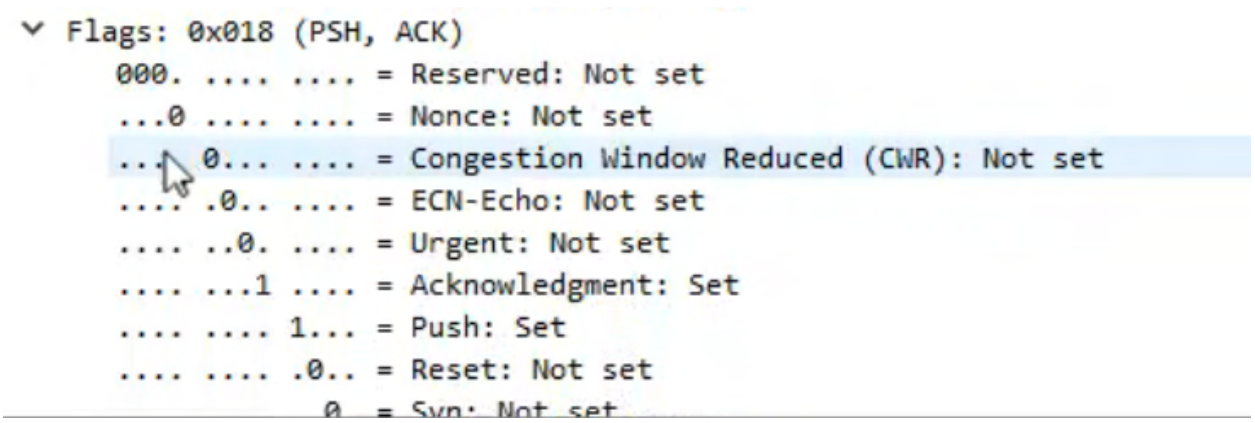


Let's analyze the data. Familiarize yourself with the way that the network application chooses their transport layer protocol whether TCP or UDP.

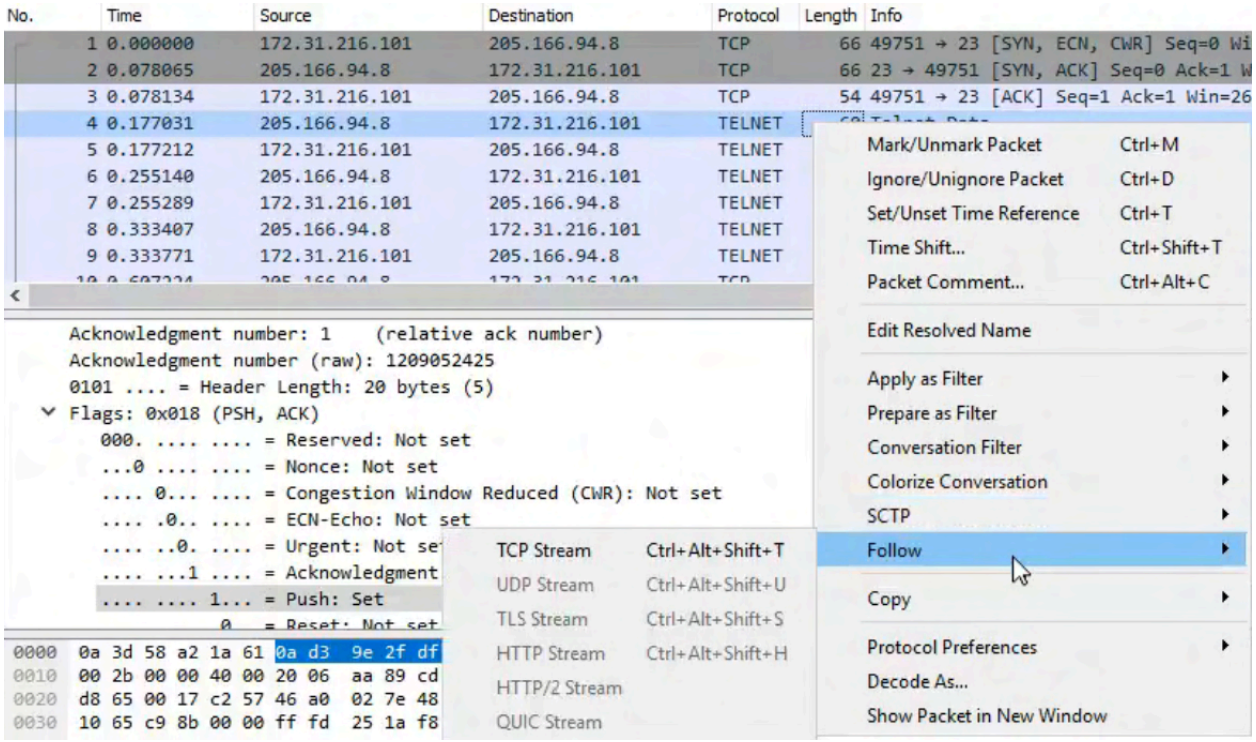
Here we can see that Telnet needs to use TCP to send keystrokes to the remote server in real time.

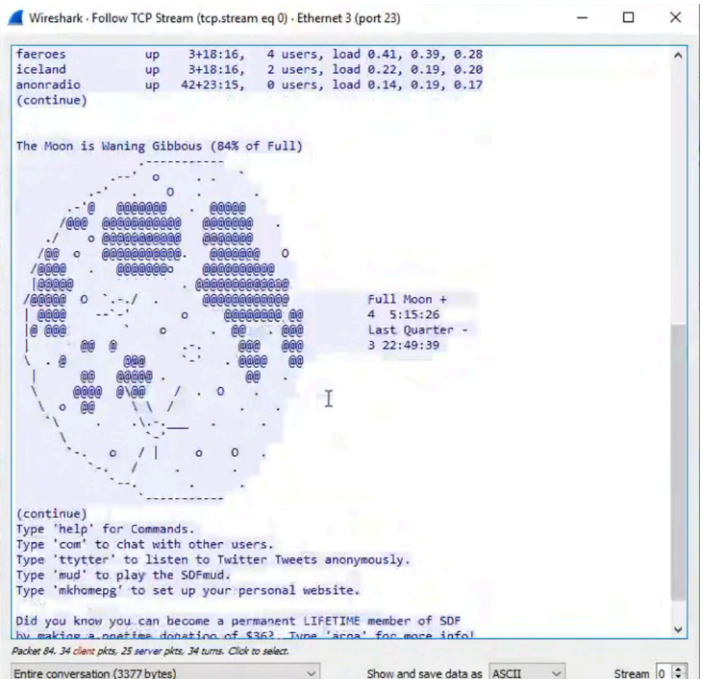
Usually real time applications uses UDP, however Telnet requires traffic to be sent reliably using a trick that would send all traffic in a NEAR real time traffic.





We can view how this works by analyzing one of the capture entries. On Wireshark we can see entry number 4 has been selected. Expanding both the Transmission Control Protocol and the Flags drop down menu, we can see that the Push flag is set to 1. This means that whenever any character comes from Telnet it will just send it. It will not wait as TCP combines a whole PDU.





We could also right click on the same entry and click on Follow -> TCP Stream

This will show everything we have seen in the Telnet session, which would mean that this is fully unencrypted and it is not secure.

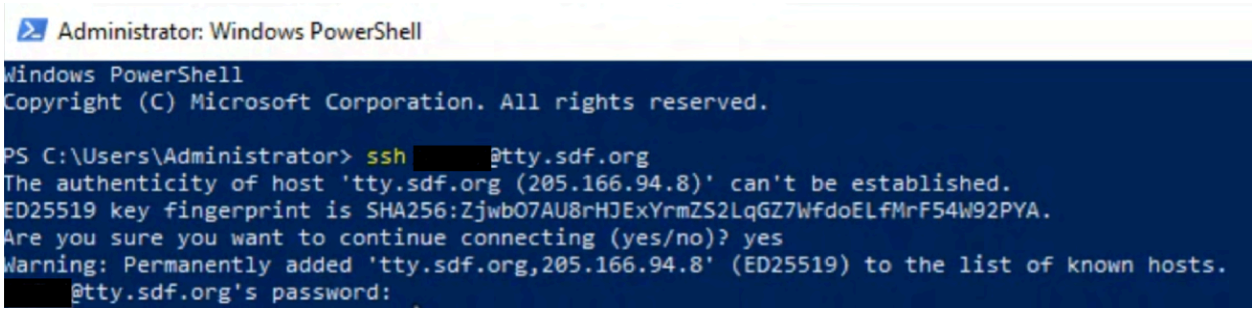
```
..%..%..%.....%.....&..... ..#..'..$..&..... ..#..'..$.....'.............x.
1....'.....ANSI.........."....."....."....."..............
sdf.lonestar.org (pts/0)
if new, login 'new' ..
```

Here is a cool example of what Wireshark is capable of showing. Here we can see both a blue and red highlight. Each highlight means something. The red highlight represents what we send to the server. The blue highlight represents what the server spits back at us.
Telnet echoes back everything back on our screen. However when inputting the password the server does not spit the information back to us which is why the password is left blank when inputting a keystroke.

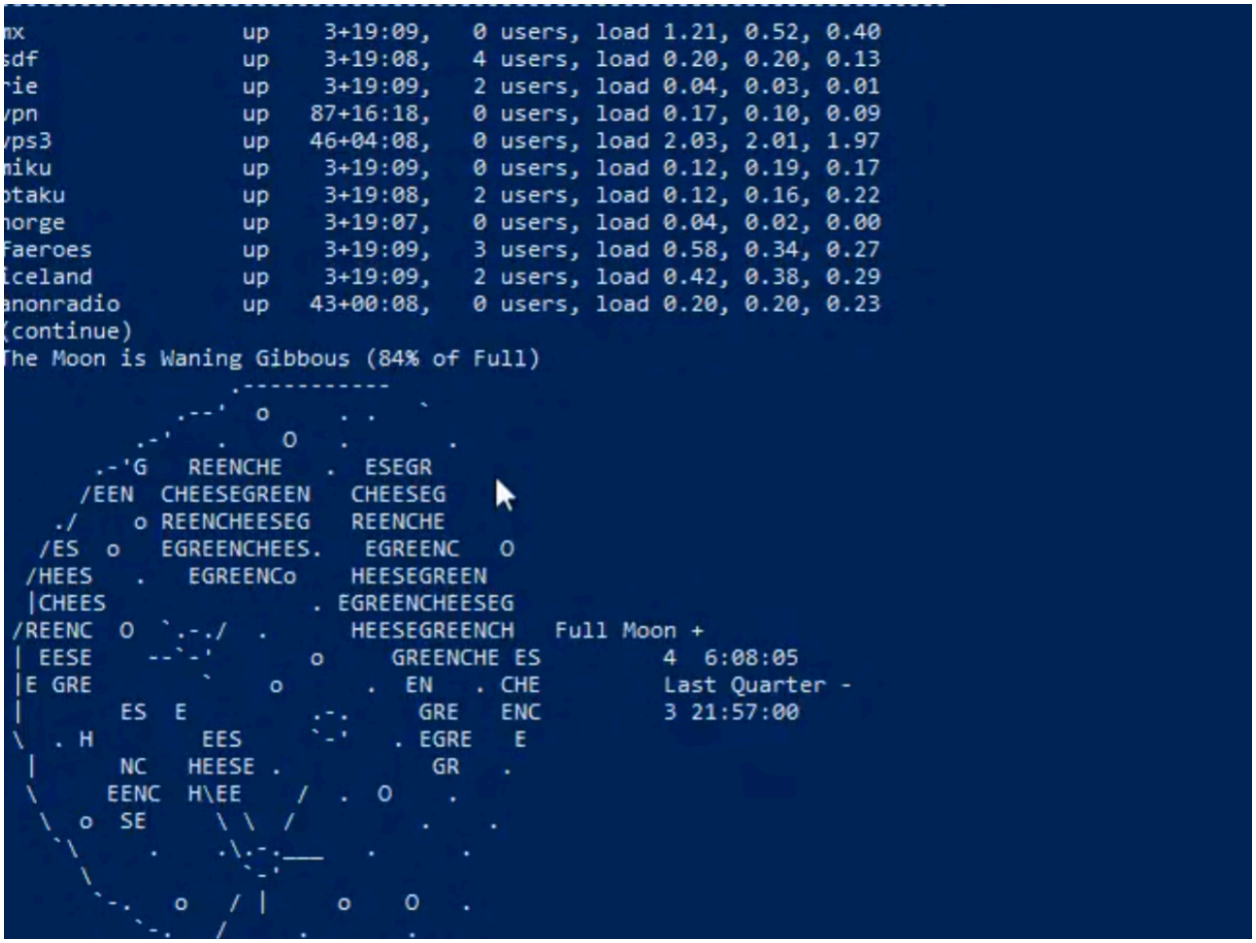
<u>Task 6 - Capturing and Analyzing SSH Sessions</u>

In this task we will be investigating encrypted protocols and traffic. Encryption means that the data or plain text is encrypted with an encryption algorithm and encryption key
This process can be viewed in its original form only if it is encrypted with the correct key
Please note that this is the most simplest form of encryption so take it with a grain of salt
Previously we have analyzed data using Telnet, much like Telnet… SSH is used to remotely access and manage a device. The key difference is that SSH uses encryption which means that SSH encrypts all data transmitted over the network.
We should note that Telnet uses `port 23` and SSH uses `port 22`
SSH also uses TLS which is the same protocol used by HTTPS to secure communication.
Please note that SSH is the predecessor of TLS which TLS being the newer standard and is more secure.

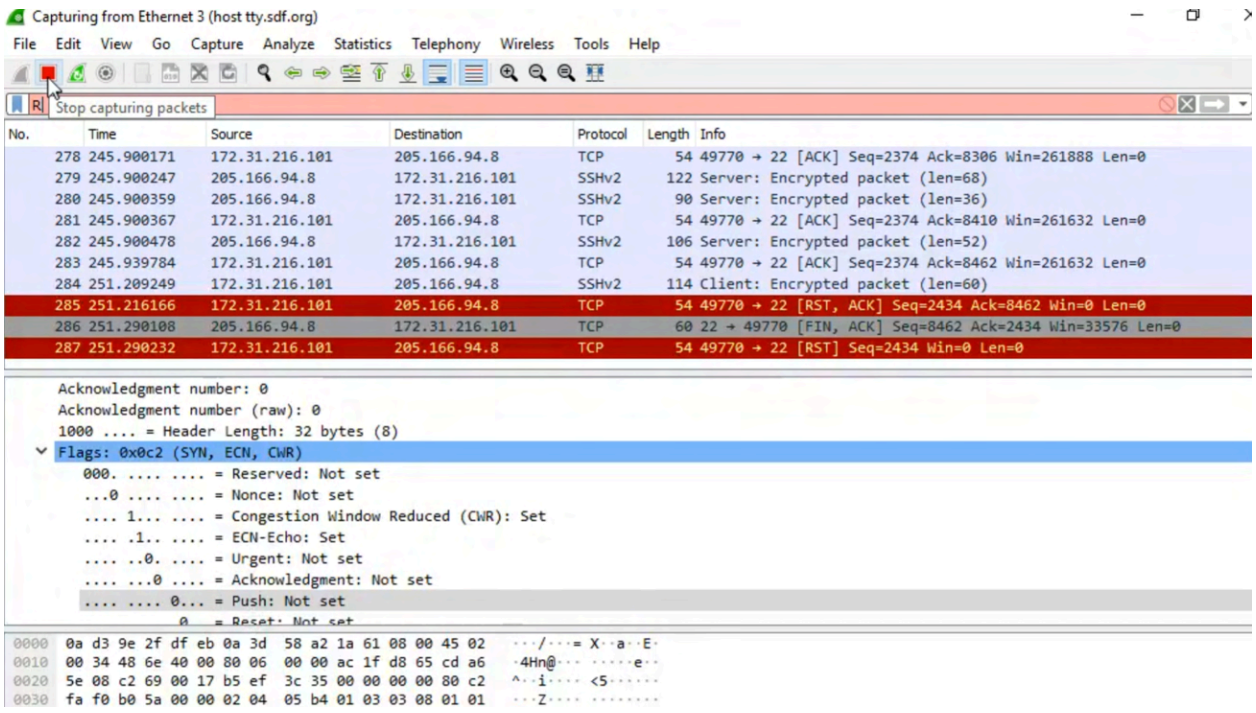
Notice how in the image above we are not capturing only SSH network packets. If we were to do so the capture filter would look something like this: `port == 22`
Instead we have inputted `host tty.sdf.org` to capture all packets regarding the host traffic of tty.sdf.org.
Before we move on, we should start capture and follow the same steps from Task 5 for telnet traffic. While finishing up telnet traffic, do not stop Wireshark Capture and move onto SSH Traffic



In the image above we have our first powershell command: ssh [username]@tty.sdf.org
The blacked out box will be the username which you will input.
When inputting your password make sure to input it correctly - it is case sensitive

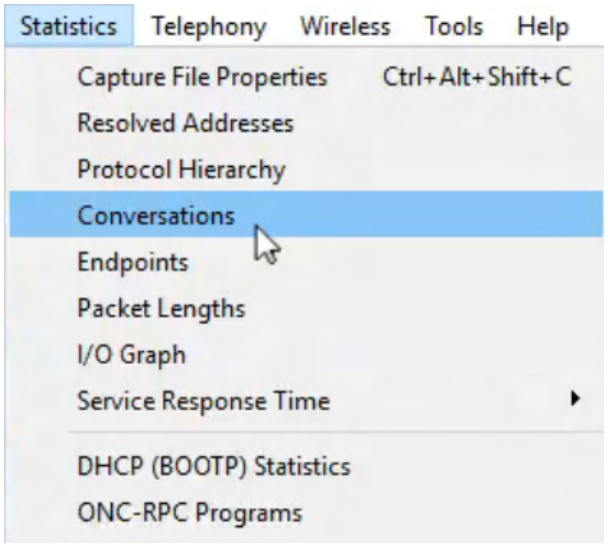


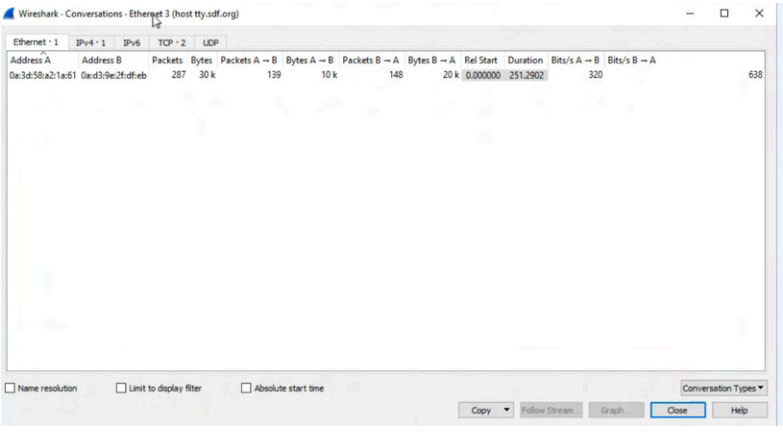
The same output will be present when using the SSH command.
We can now exit powershell and go back to Wireshark



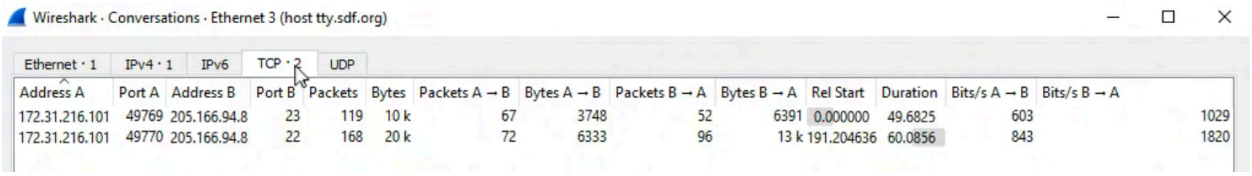
When exiting powershell - there will be an error message on Wireshark that will be captured This is not an actual error, rather it is SSH being cautious. Wireshark is sending out a message that when your trying to connect to an unknown host, it will give you a warning. In this capture you are able to identify the ip address as well as its own fingerprint, in the real world you will be comparing the fingerprints with the fingerprints you were given in order to determine whether you are connecting to the correct host instead of an imposter.

Interestingly, Wireshark is also able to capture how many conversations were taken in place in the current entry.

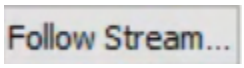


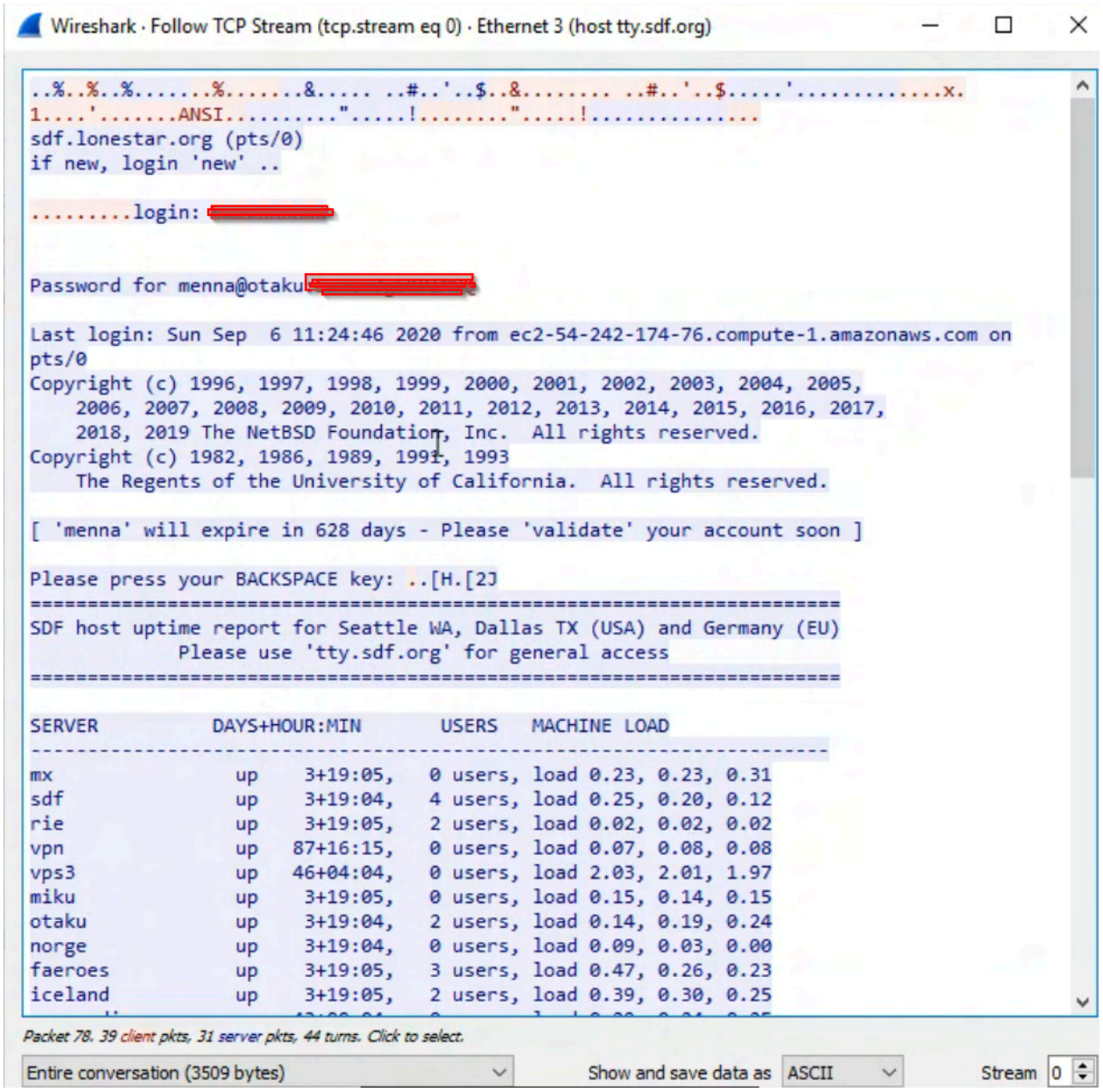


When going into statistics -. Conversations, you are able to find multiple tabs with options Thernet, IPv4, IPv6, TCP and UDP
Since both SSH and Telnet uses TCP we can click on TCP



We can note that we are able to see 2 different entries, We can also note that although the IP address is the same they are using two different ports, 22 and 23
If we click on the first entry with the port number of 23 and hit follow stream





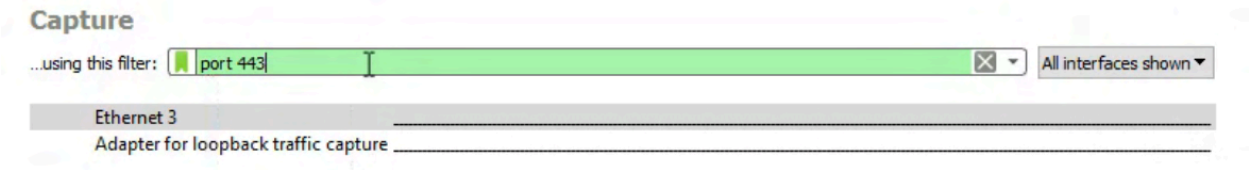
We can see that the whole conversation is visible, which is not secure. We can note that in the event of a Man in the Middle attack were to occur, they would be able to see everything that happened within this conversation.

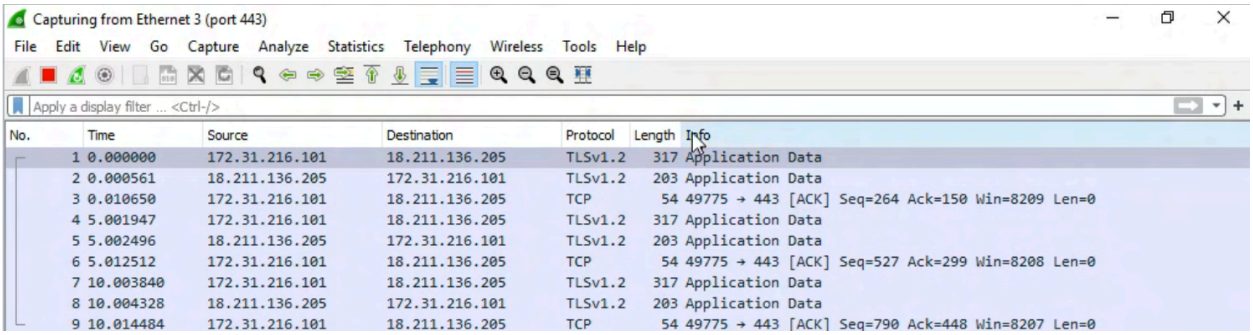
However, lets view what happens when we enter into the entry with the port number of 22 instead.

Notice how everything here is encrypted! This is the same entry as before… though this time a man in the middle attack will be quite difficult to decrypt this message.

Task 7 - Generate, Capture, analyze ten decrypt HTTPS Traffic

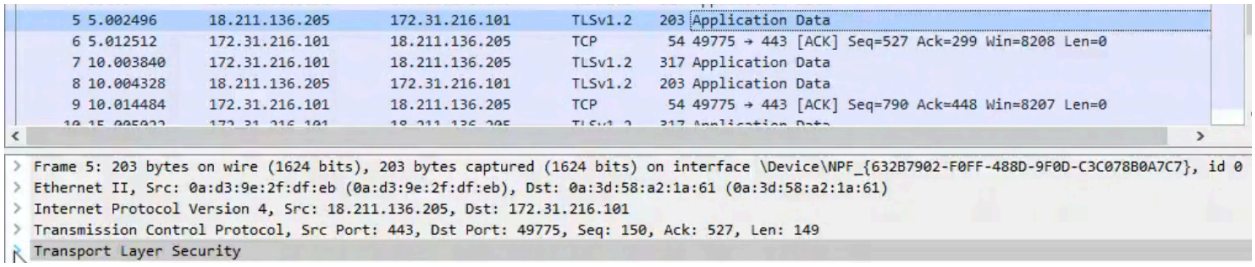
In this task, we will be capturing packets from HTTPS Traffic. We can note that HTTPS is basically HTTP that operates with TLS.
HTTP operates on port 80 while HTTPS operates on port 443.
We can begin Wireshark capture using capture filter of port 443

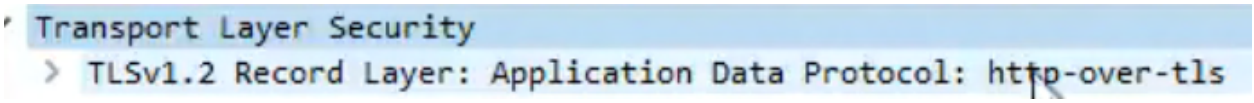




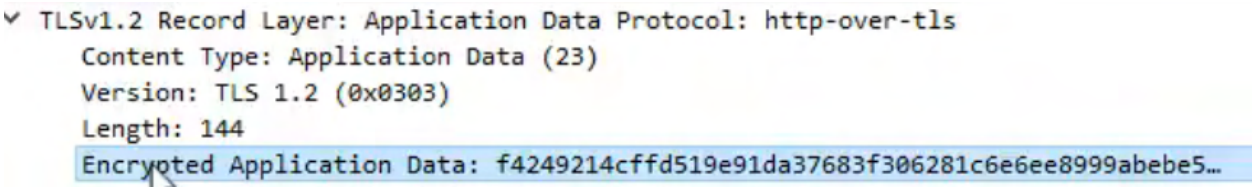
There is already traffic that is being captured using this filter. If there aren't any traffic being produced, which is odd, you can open google chrome and browse on the web for anything. There should be network traffic after following that step.

On entry 5, we can click on this application data and expand the transport layer security tab on the bottom

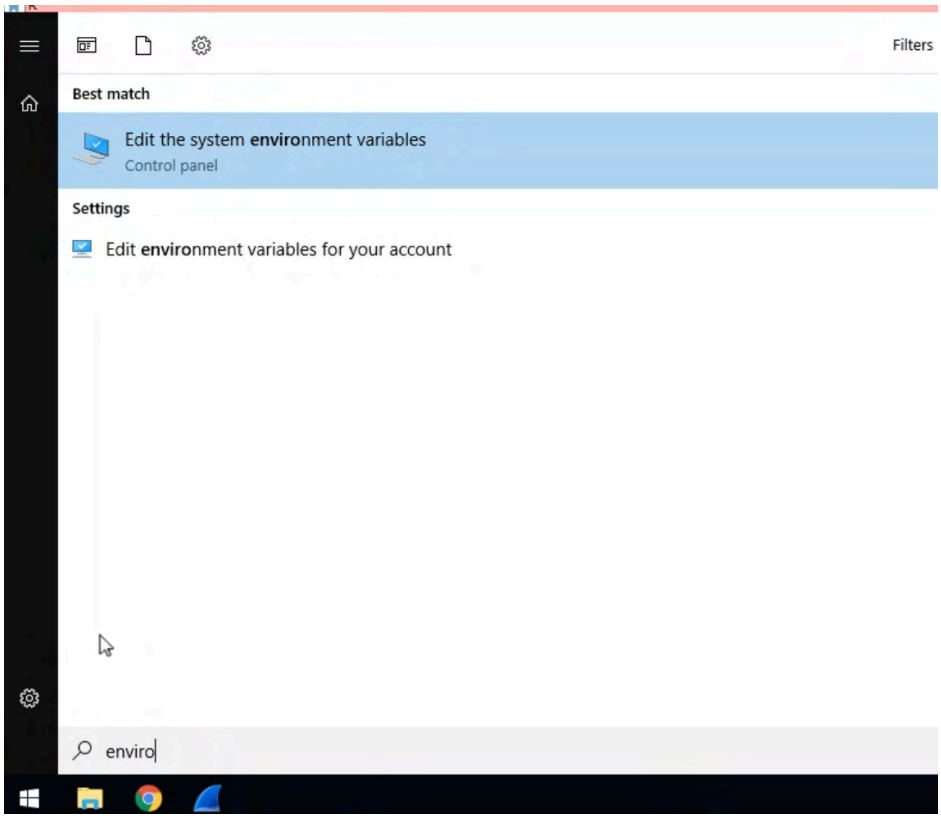


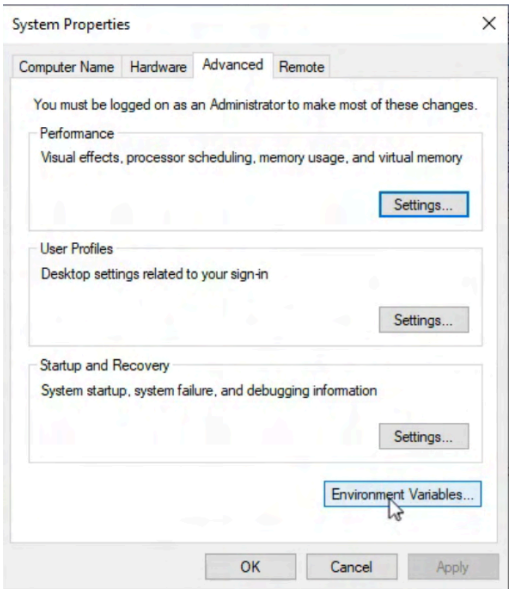


We can see that Wireshark is informing us that HTTPS is HTTP over TLS.

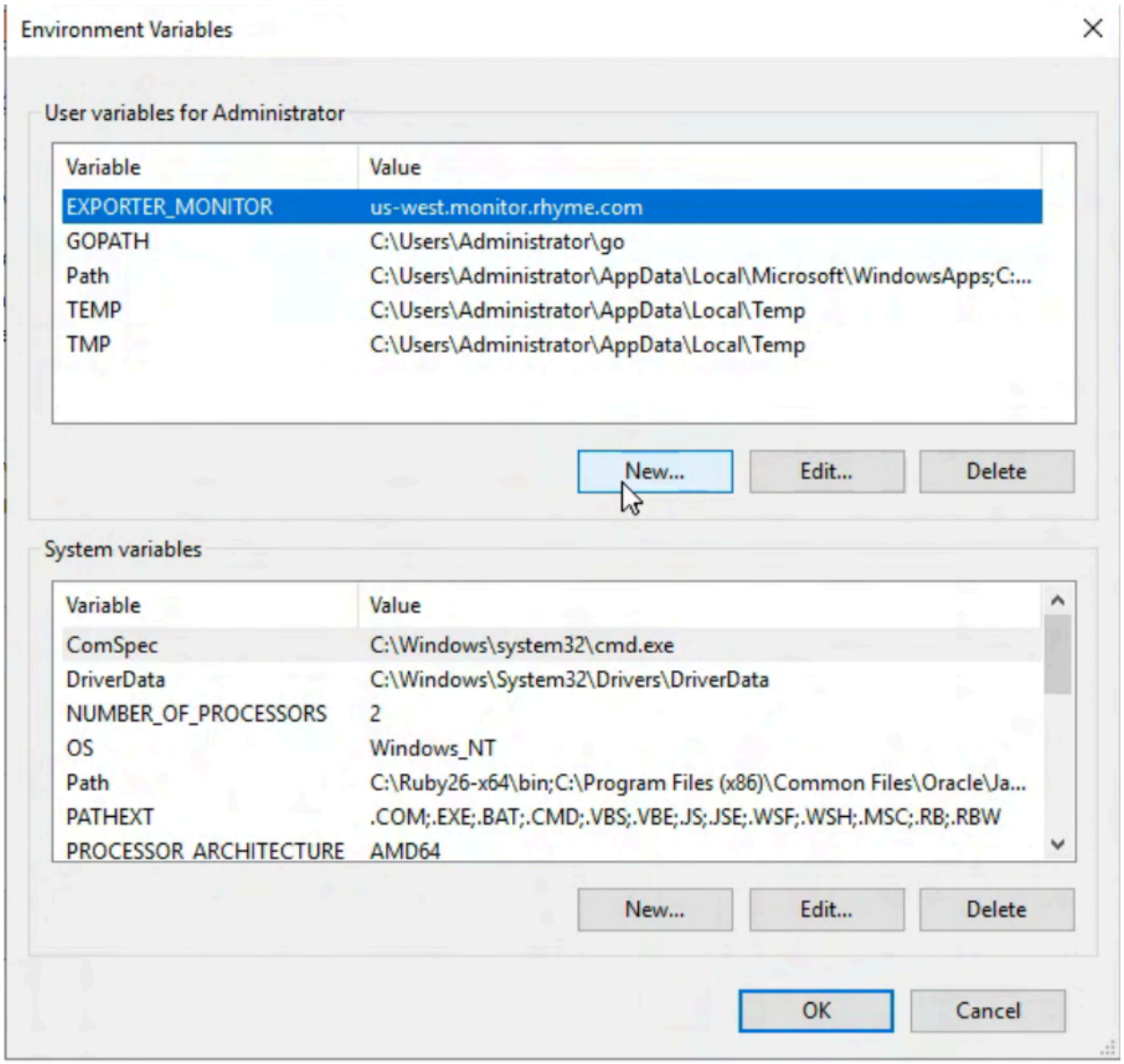


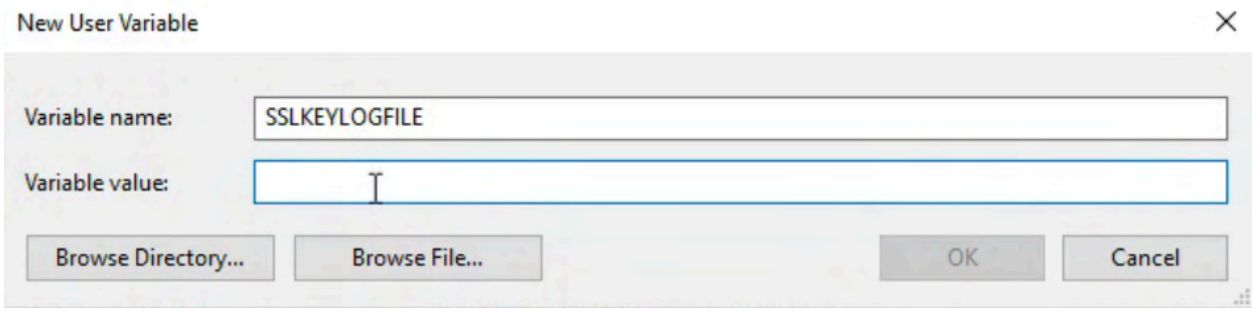
When expanding even further, we can notice that Wireshark is informing us that this particular entry is being encrypted. We won't be able to decrypt this data due the fact that we do not possess the private key used for encryption.
However, say that you need to decrypt this data, perhaps for network administration or testing something in your home lab. We can decrypt SSL traffic using a pre-master secret key.
To set up this, we must first go in to the windows button and search for environmental variables

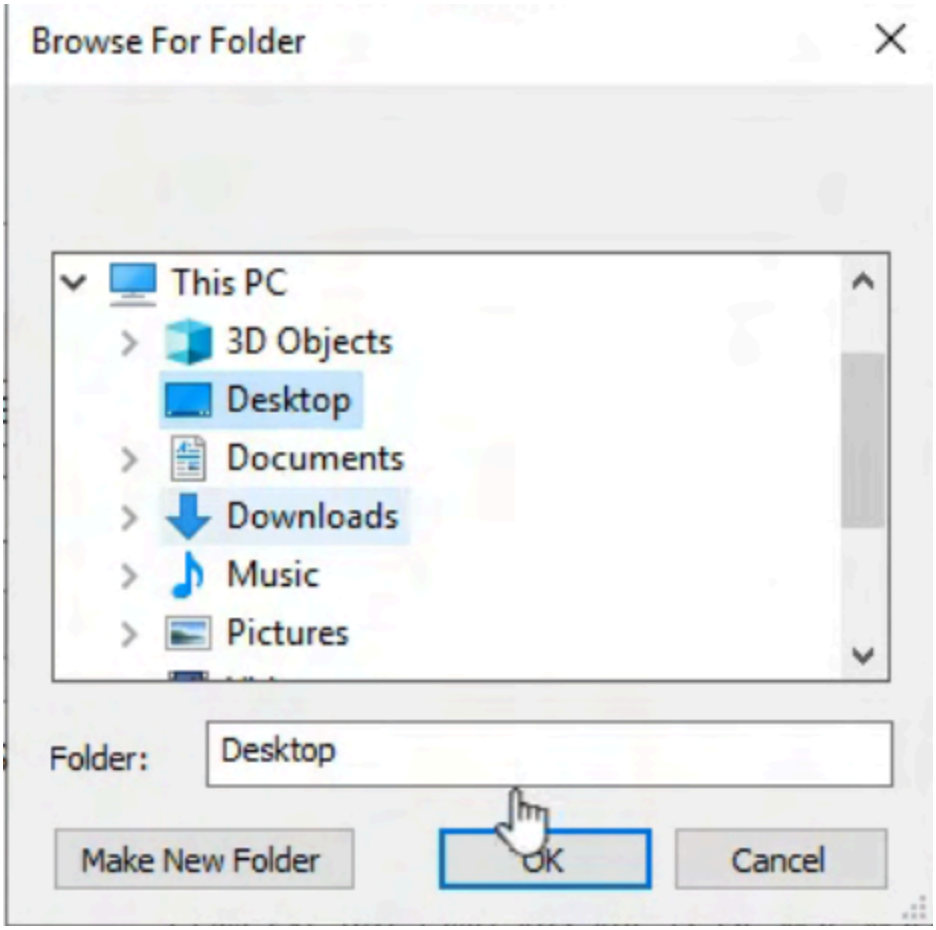


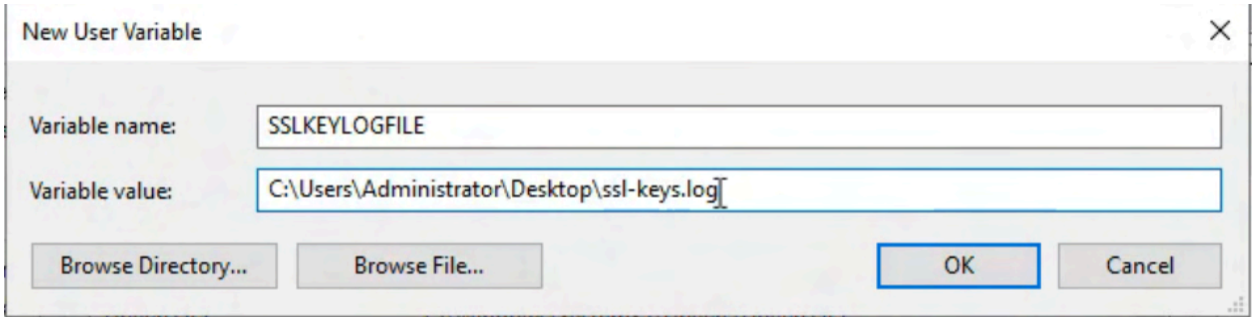


In here you will be able to click on the button Envrionmental Variables.
Once in the environmental variables tab we should be able to add a new variable


Environment Variables
User variables for Administrator
Variable
Value
EXPORTER_MONITOR
us-west.monitor.rhyme.com
GOPATH
C:\Users\Administrator\go
Path
C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps;C:...
TEMP
C:\Users\Administrator\AppData\Local\Temp
TMP
C:\Users\Administrator\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
2
OS
Windows_NT
Path
C:\Ruby26-x64\bin;C:\Program Files (x86)\Common Files\Oracle\Ja...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.RB;.RBW
PROCESSOR_ARCHITECTURE
AMD64
New...
Edit...
Delete
OK
Cancel


New User Variable
Variable name:
SSLKEYLOGFILE
Variable value:
Browse Directory...
Browse File...
OK
Cancel

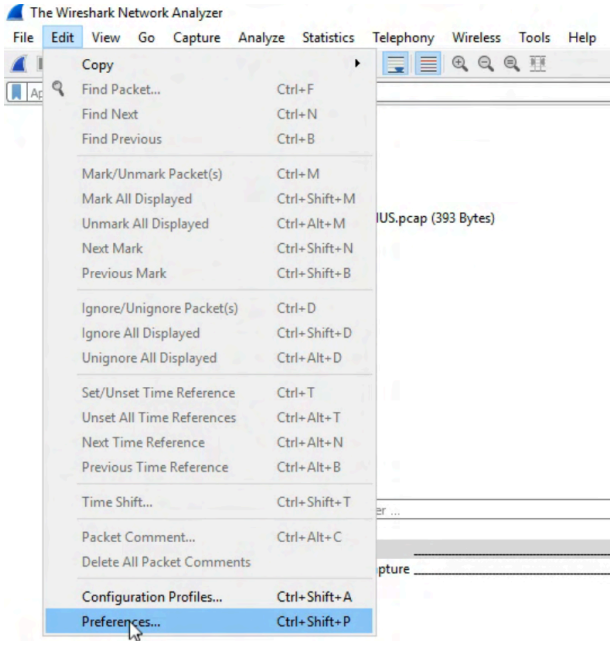


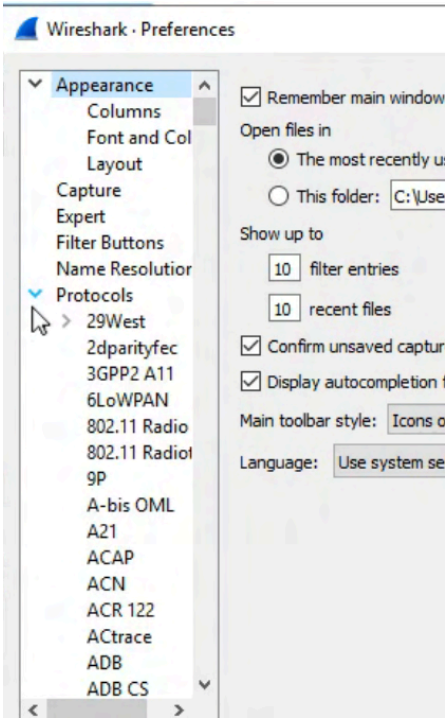


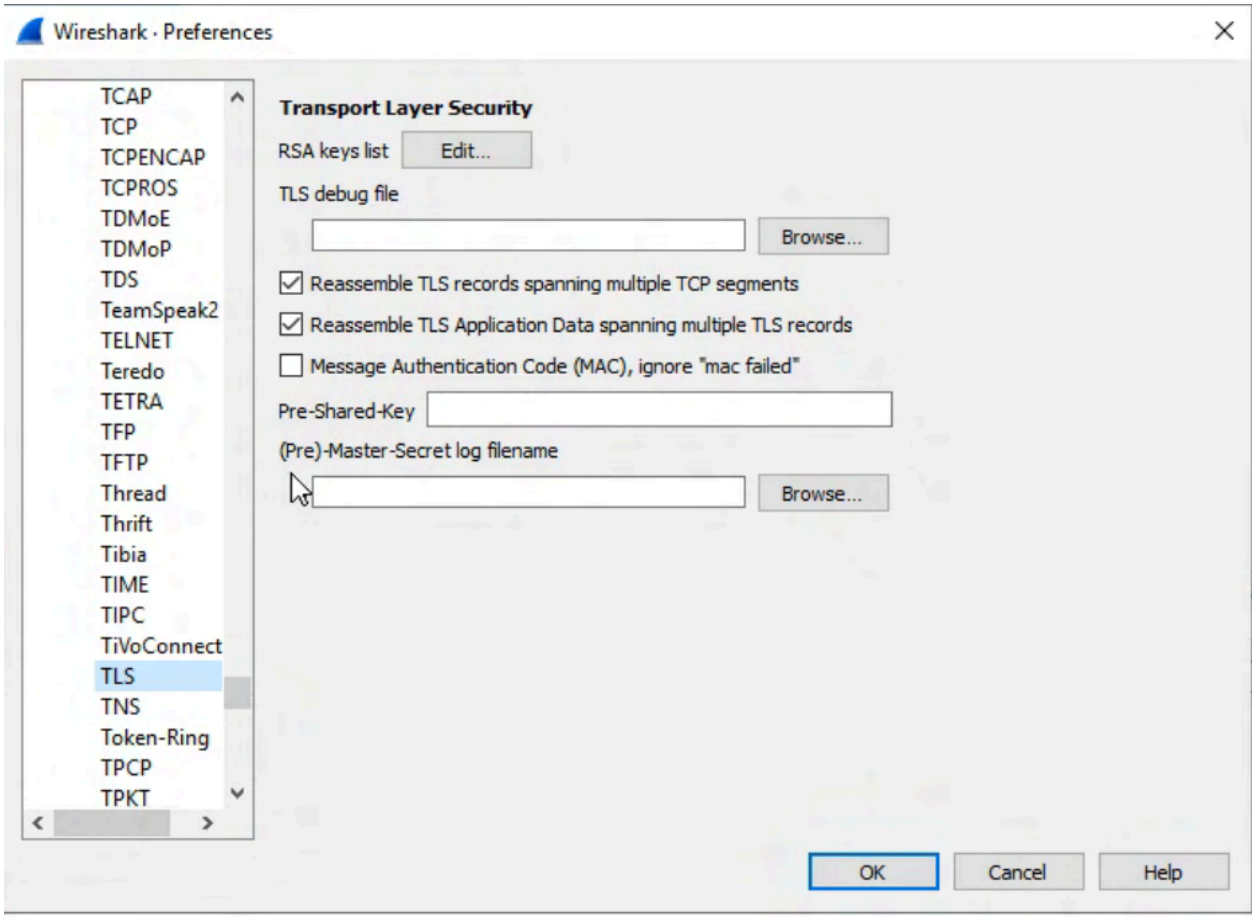
In the images above we are able to see that we creating a new variable named SSLKEYLOGFILE
This will be routed through the value C:\USers\Administrator\Desktop
We will be creating a new file using the back-slash key “\”
We are going to be creating a file ssl-keys.log
When finishing this, we should be able open google chrome and search for any random website such as amazon.com

When completing these steps we will be able to see the new file created as as shortcut on our desktop home screen.

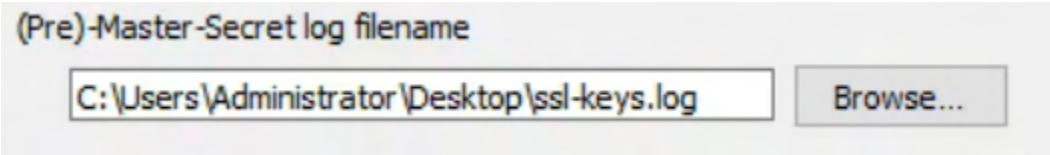
We will now return to Wireshark and go into Edit -> Preferences -> Protocols -> TLS



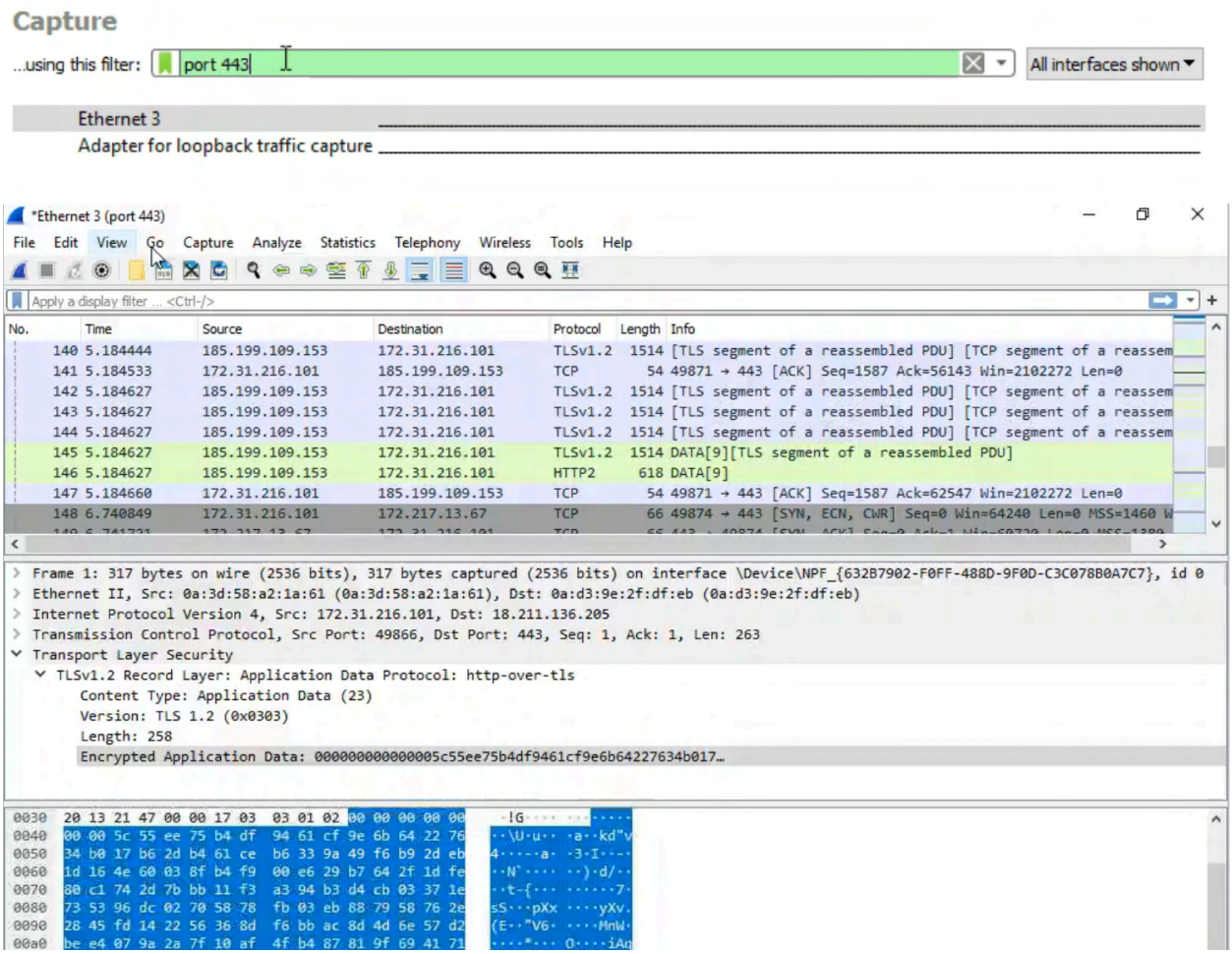




When competing the steps shown in the above image, the (Pre_-Master-Secret log filename can be found. Here we will browse the desktop and find the file name we just made. It will look something like this

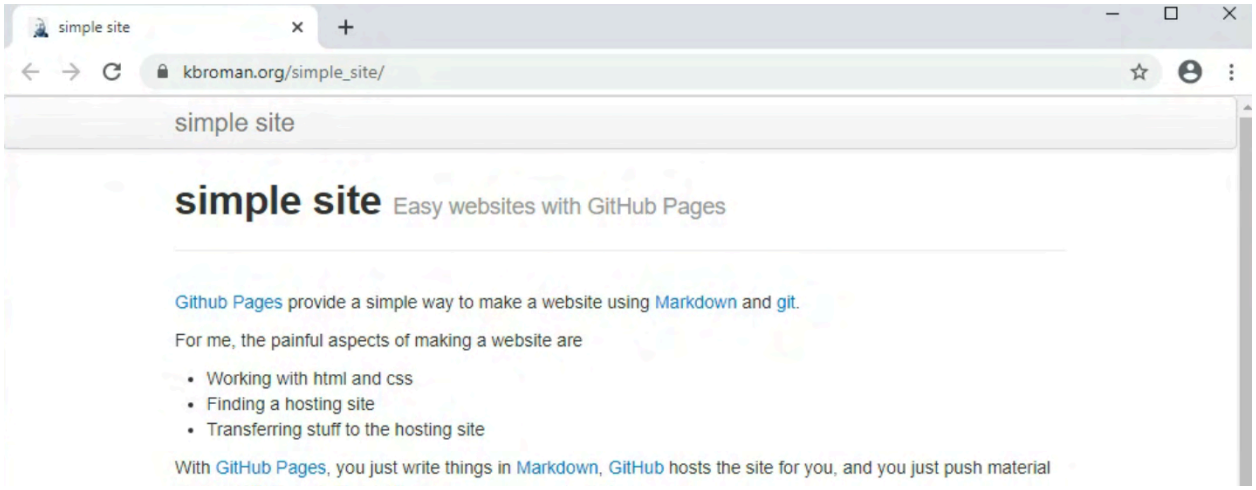


When completing these steps we should be able to capture network packets and view SSL in a decrypted format

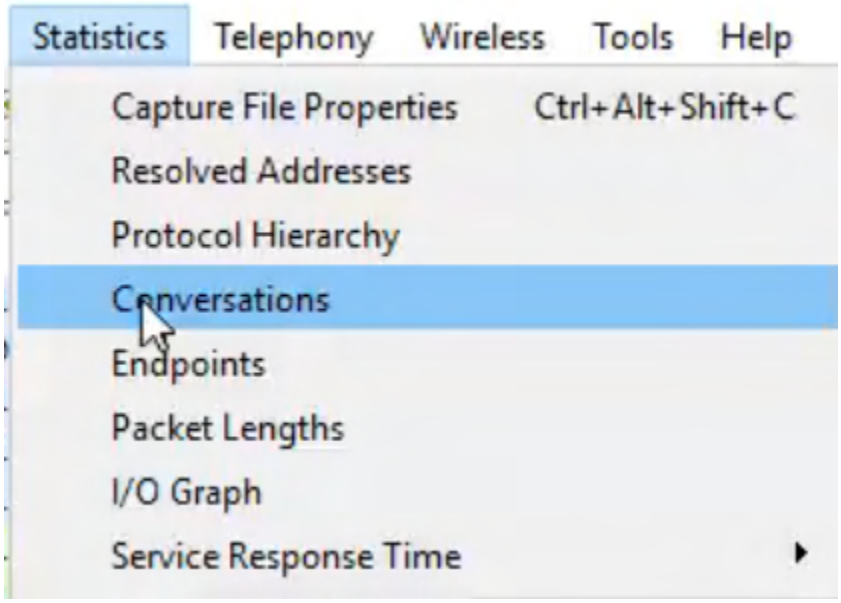


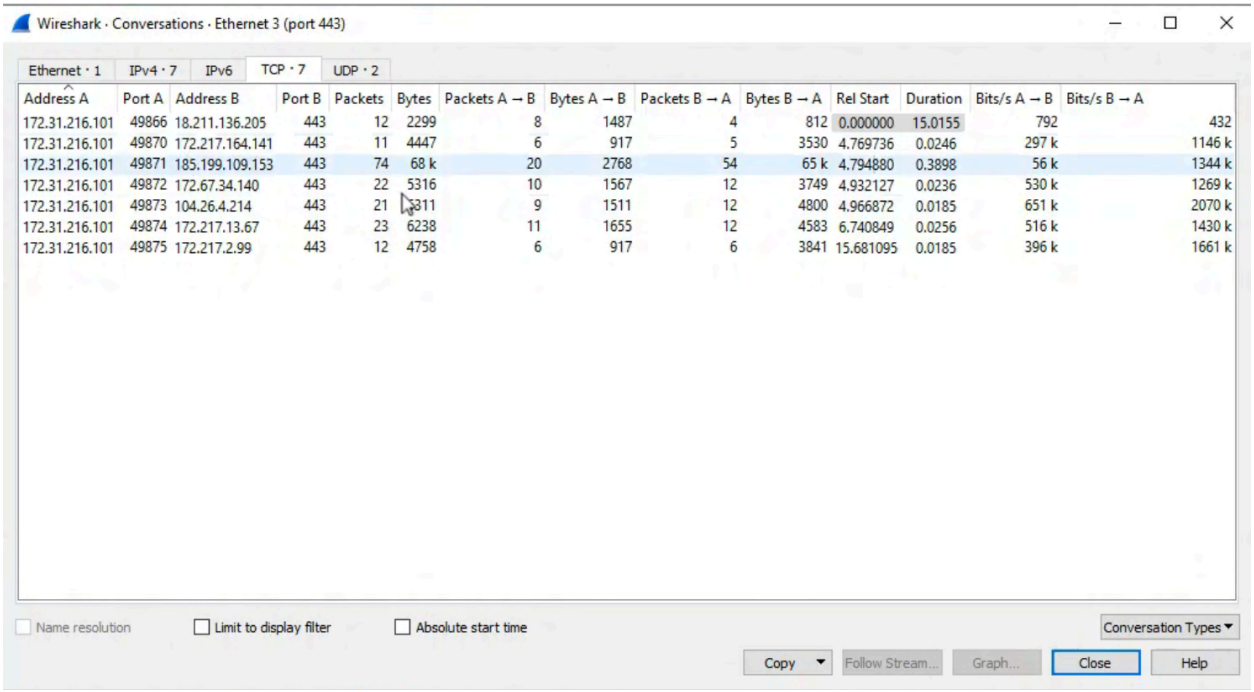
Notice how we have begun the network traffic using the capture filter port 443
When entering these formats we can select an entry and inspect it further

We will be capturing network traffic from this website “Simple Site”

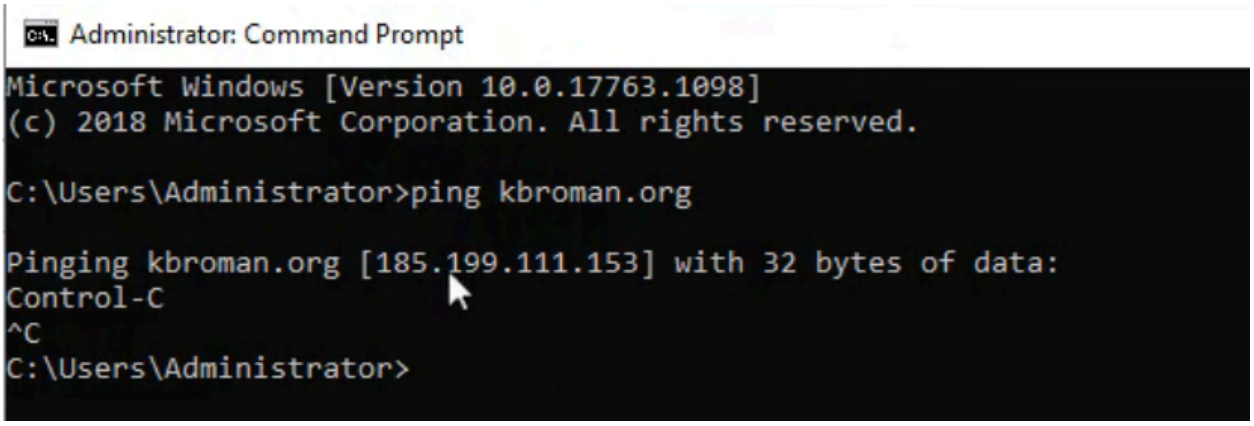


It should look something like this. With this we should be able to stop Wireshark capture. We will then head on over to statistics tab and click on conversations





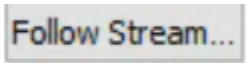
When on this page it will be a little difficult to find the correct entry to inspect further… We can mitigate this by pinging the server in cmd.

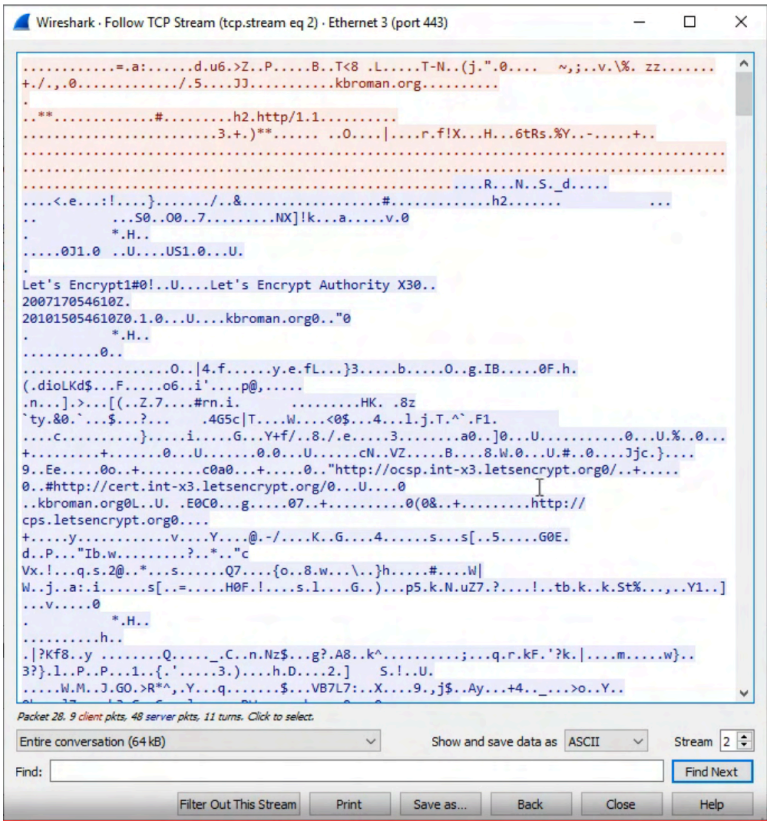


It should look something like this. We can note that the IP address is shown so we can use this information to narrow down the search in our entry list on Wireshark.

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
172.31.216.101	49866	18.211.136.205	443	12	2299	8	1487	4	812	0.000000	15.0155	792	432
172.31.216.101	49870	172.217.164.141	443	11	4447	6	917	5	3530	4.769736	0.0246	297 k	1146 k
172.31.216.101	49871	185.199.109.153	443	74	68 k	20	2768	54	65 k	4.794880	0.3898	56 k	1344 k

We can see the IP address is listed here. We can click on it to have it highlighted and head on over to the follow stream button





We can see that on the image to our left, we have the encrypted… however before we were only able to see application data. Now we are able to see everything else like we normally would on HTTP traffic. We are even able to see the source code for this website.

Lessons Learned

Using Wireshark and most of its basic functionalities: capturing some network traffic that is flowing through your machine now and analyzing already captured network traffic by opening a stored capture file.

Generating and capturing RADIUS Traffic, analyzing it and viewing it in Wireshark. Knowing what the RADIUS Architecture consists of and decrypting the encrypted password with the shared secret using Wireshark.

Knowing the basics about HTTP, and knowing the difference between Wireshark's Capture and Display Filters. Connecting to an HTTP Server and initiating Basic HTTP Authentication and capturing its Traffic on Wireshark, analyzing the captured packets and seeing the username and password being sent.

Initiating an HTTP Form-based authentication, capturing it in Wireshark and analyzing it so you can see the username and password clearly. Also Capturing DNS Traffic.

Knowing how Telnet works, starting a Telnet Session with a remote Device using Powershell, capturing its traffic in Wireshark and analyzing it from the Security perspective

Opening a SSH Session with the same Device as in Task 5, capturing the traffic and comparing it with the Telnet Packet Capture. Capturing traffic based on the host involved and how to see all the host, packet and protocol statistics as well as conversations that have happened in a certain Capture.