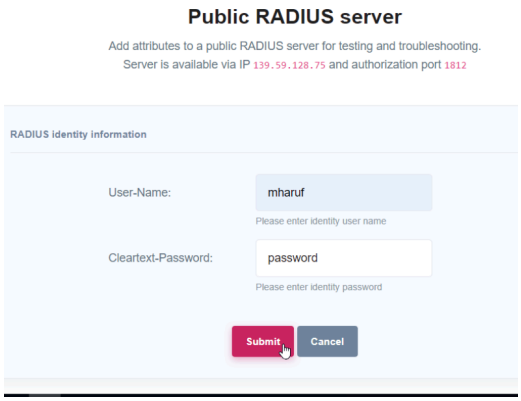
Analysing network traffic with Wireshark

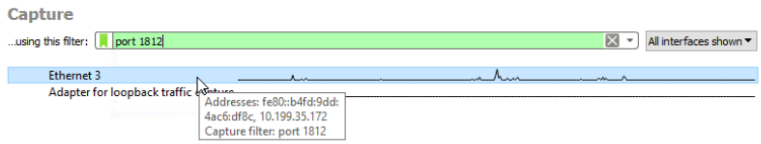
In this project I was able to utilise many features of Wireshark to capture, filter, analyse, and decrypt packet captures. The work was completed in a virtual machine provided by the course.

Generating, capturing and analysing RADIUS traffic

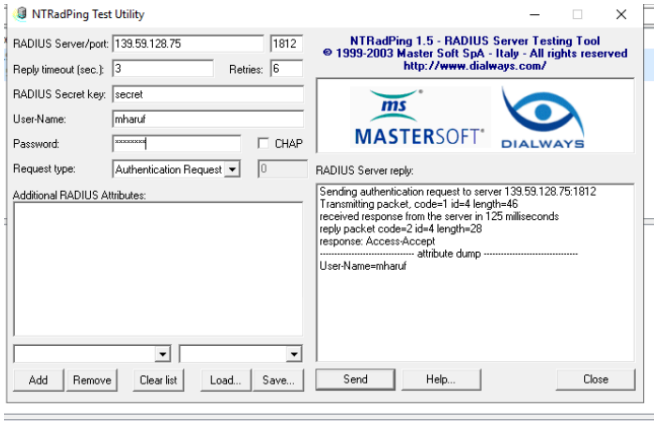
To begin with, I needed to generate some RADIUS traffic.



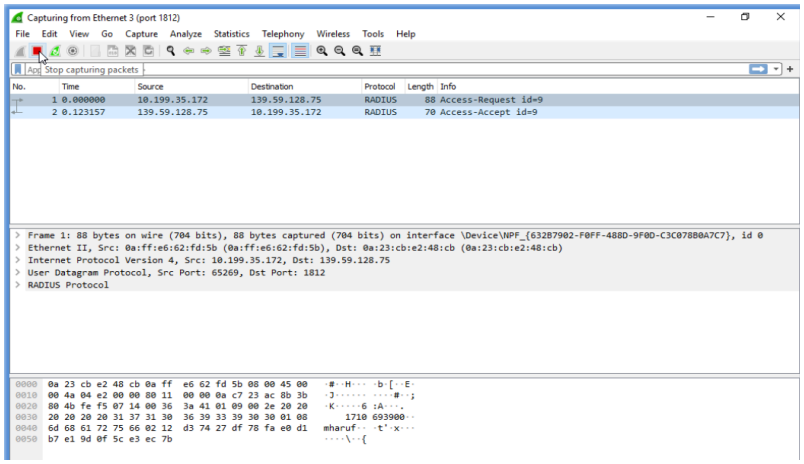
To do this, I used a public RADIUS server to input a username and password while noting the IP address and port number.



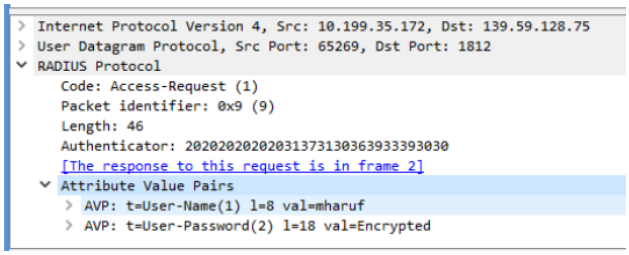
I then used the filter bar on Wireshark to filter for port 1812, as this was the port indicated earlier on the public RADIUS server, so that the only packets that are captured will be of the RADIUS protocol. I then started capturing packets by double clicking the interface ‘Ethernet 3’. Following this, I needed to ping the RADIUS output using NTRadPing Test Utility.



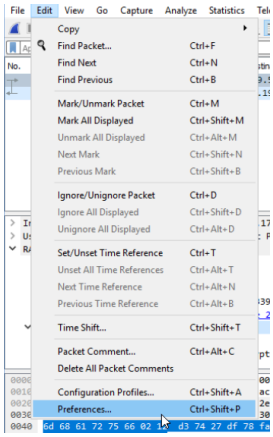
On this page I inputted the IP address indicated earlier on the RADIUS public server as well as the port number. Next, I inputted the same username and password I had used on the public server. I left everything else as it was including the secret key. I then clicked ‘send’.



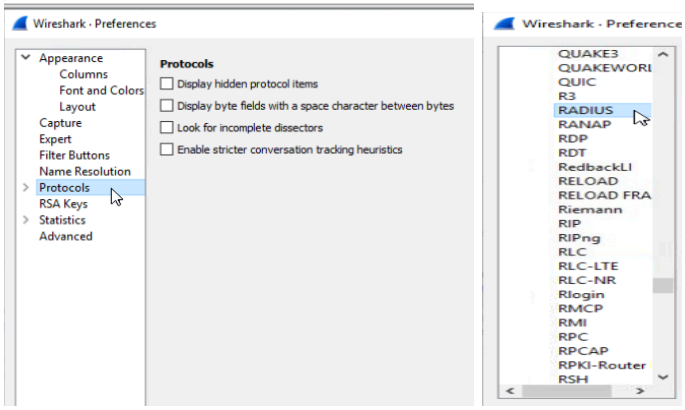
After sending the ping, Wireshark captured packets detailing the authentication process.



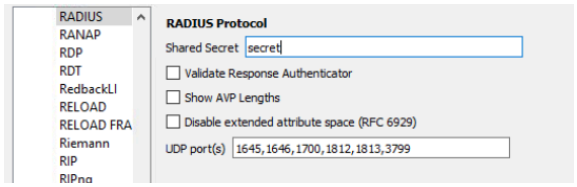
By clicking on the packet capture for the ‘Access-Request’ and navigating to the ‘Attribute Value Pairs’ section, I was able to see the value of the username as shown in the screenshot above. However, the password was encrypted. I now needed to decrypt the password using Wireshark.



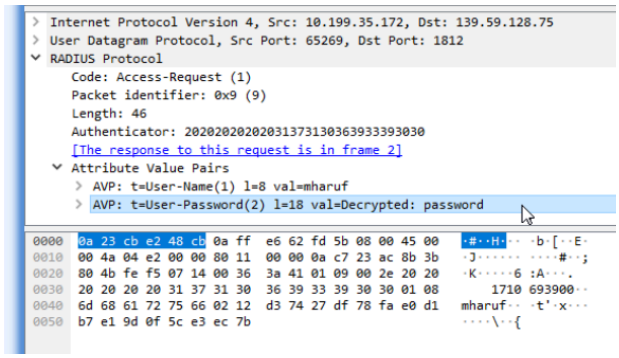
I clicked on ‘Edit’ then ‘preferences’.



This opened the window displayed above. I then selected ‘Protocol’. This opened a list of protocols from which I chose ‘RADIUS’.



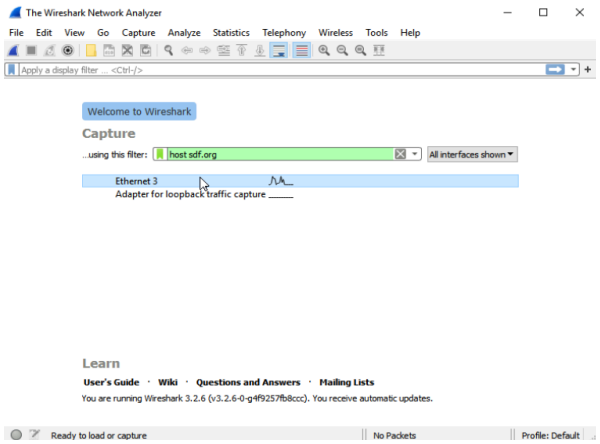
This opened a page where I was able to input the secret key from earlier, in order to decrypt the password. I then pressed ok.



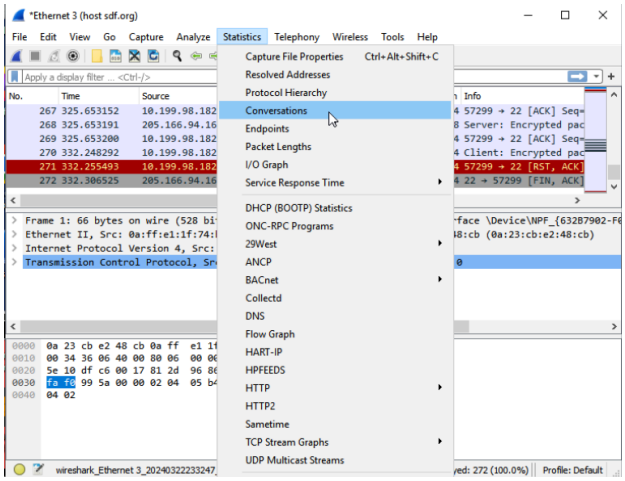
As displayed in the screenshot above, the password has now been decrypted and is now visible in the packet captures ‘Attribute Value Pairs’ section.

Generating, capturing and comparing Telnet output with SSH output

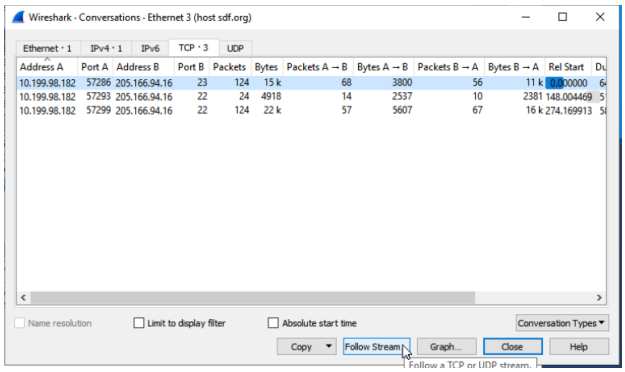
In this task I followed instructions to start a Telnet session with a remote device using Windows Powershell. I then started an SSH session in the same way. Beforehand, I had started a packet capture filtering for these sessions using the host name ‘sdf.org’, as shown in the screenshot below.



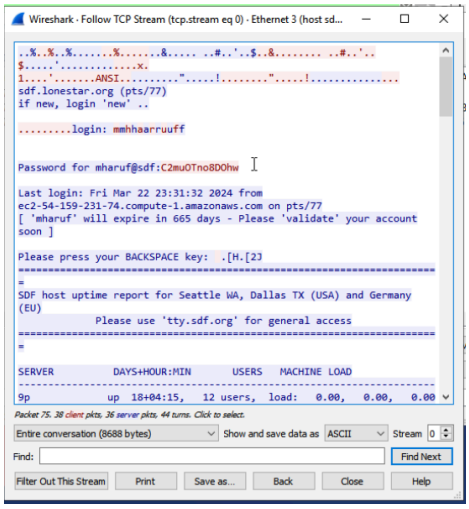
After generating the output from the Telnet and SSH sessions, I stopped the packet capturing on Wireshark.



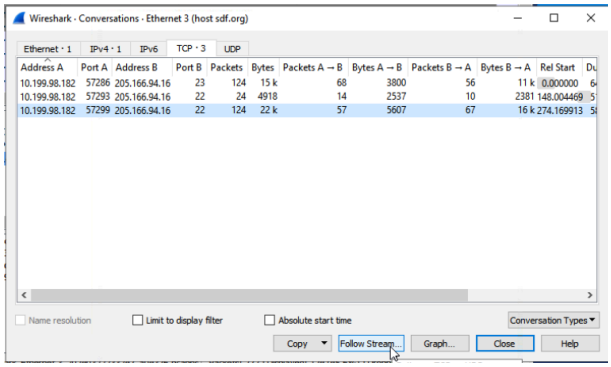
I now needed to compare the output for the Telnet session and the SSH session. To do this, I clicked on ‘Statistics’ then ‘Conversations’ on Wireshark, as shown in the screenshot above.



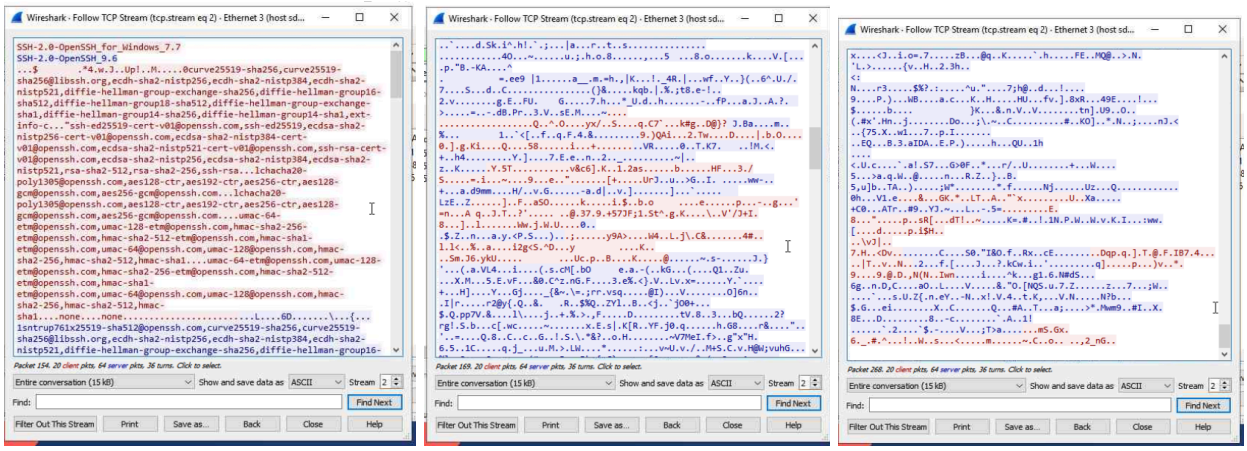
This opened up the page shown above. To view the conversation that took place during the Telnet session I clicked on the header ‘TCP - 3’ before selecting the first row. I then clicked on ‘Follow Stream’.



As shown in the screenshot above, the data was not encrypted in the Telnet session because I am able to view the login and password that I had entered during the session.



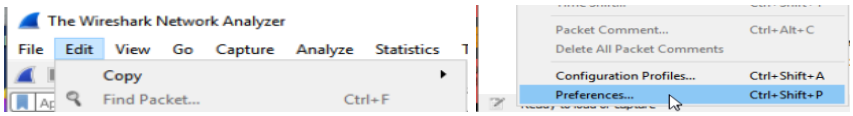
I then went back to the conversations page from earlier as displayed above, and clicked on the third row, which is the SSH session that I ran earlier. (Second row was a mistake during the SSH session). I then Clicked on ‘Follow Stream’.



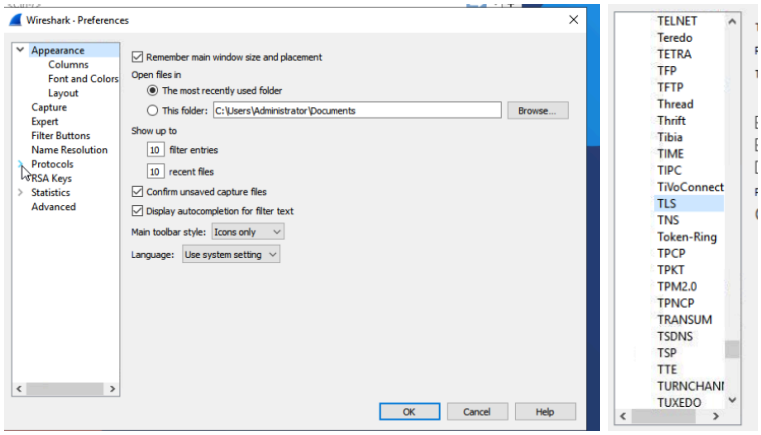
As shown in the screenshots above, the information from the SSH session was fully encrypted. This is because SSH is the secure version of Telnet in the same way that HTTPS is the secure version of HTTP. With this task, I was able to visually see the difference between the two protocols.

Generating, capturing, analysing and decrypting HTTPS traffic.

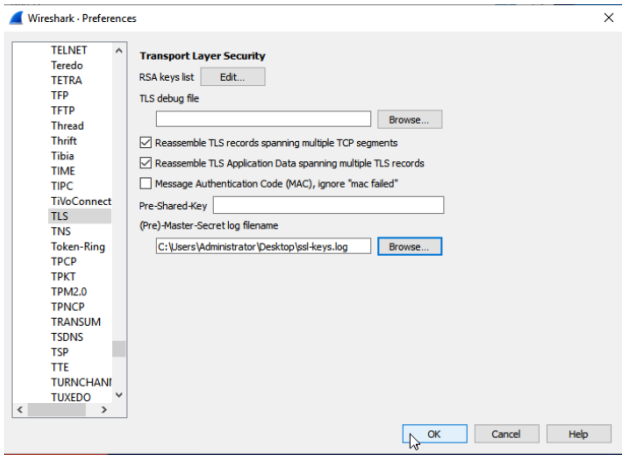
Since HTTPS is a secure protocol I need to decrypt it in Wireshark. To do this I needed to use the pre-master secret key method. I needed to receive the secret key from the web browser (client), before logging it in an SSL key log file. After completing these steps as instructed, I now needed to input the pre-master secret key into Wireshark so that when I capture the HTTPS traffic, I will be able to decrypt it using this key.



To do this I clicked on ‘Edit’ then ‘Preferences’ on Wireshark before capturing any packets. This is displayed in the screenshots above.

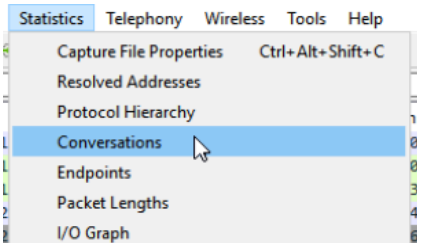


This opened up the page shown in the first screenshot above. I expanded the ‘Protocols’ tab before scrolling down to find and select ‘TLS’ as displayed in the second screenshot above.

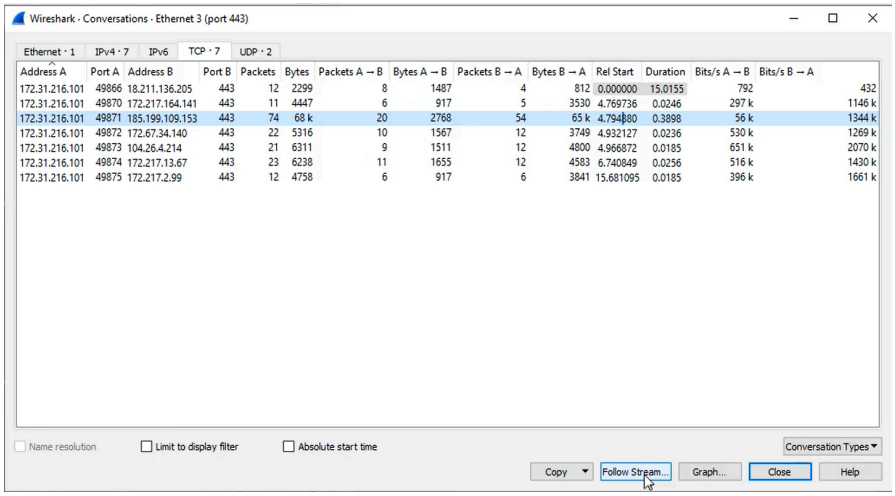


This opened up the page displayed above. I then needed to input the SSL key log file that I had created earlier which contained the pre-master secret key. I did this by selecting ‘Browse’ under ‘(pre)-Master-Secret log filename’ as shown in the screenshot above. I then navigated to the secret key and selected it before clicking ‘OK’ on the page shown above.

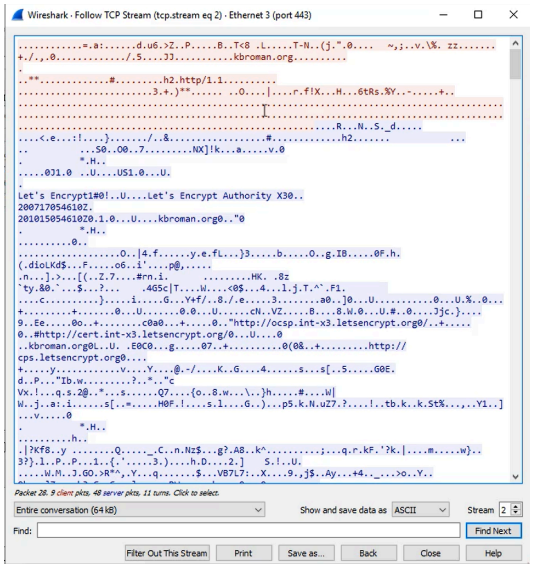
At this point I had informed Wireshark of the pre-master secret key and was now ready to start capturing HTTPS traffic. To do this I inputted the filter ‘port 443’ (which HTTPS operates on) on Wireshark before clicking on the interface ‘Ethernet 3’ to begin capturing. This is displayed in the screenshot above. Once the capturing process had begun, I went onto Google Chrome and searched for ‘kbroman.org/simple_site/’ to generate some HTTPS traffic. I then went back to Wireshark and stopped the capturing.



I now needed to see if the pre-master key I had inputted into Wireshark earlier had successfully decrypted the HTTPS traffic. To do this I clicked on ‘Statistics’ then ‘Conversations’ as shown in the screenshot above.



This opened up the page displayed above. I saw that there were multiple different conversations that were captured, so I needed to identify which server I had connected to when I used Chrome. To do this I pinged the host name of the site I visited using the command prompt on Windows. This led me to the IP address of this particular site, so that I could identify which conversation in the screenshot above was from the site I visited. I realised that it was the third row. So I clicked on the third row before clicking ‘Follow Stream’.



This opened up the page displayed above showing that the premaster key had successfully decrypted the HTTPS traffic that was captured.

Summary

Completing this project has given me great insight into the various capabilities of an important cyber security tool. I was able to capture, filter, and analyse different types of traffic such as RADIUS, Telnet/SSH, and HTTPS. I was also able to see the difference between encrypted and unencrypted packet captures. Additionally, I learned how to view packet and protocol statistics on Wireshark. Furthermore, I learned how to decrypt packet captures using Wireshark. This experience will be useful when analysing network activity in future cyber security roles.