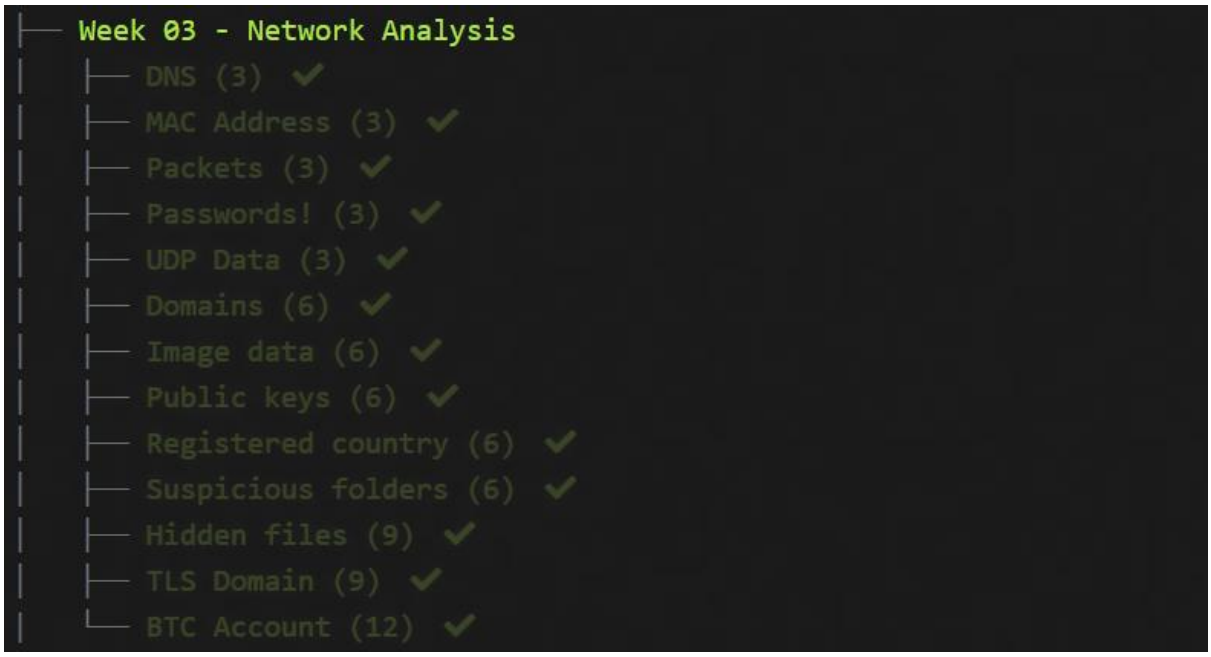
CYBERCRIME

Author: Kharim Mchatta

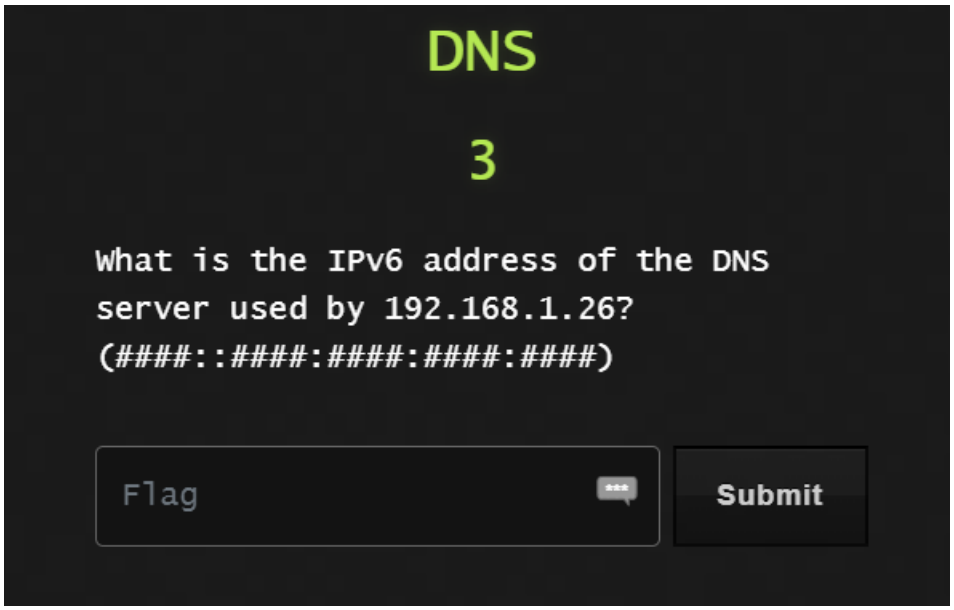
Title: Africa Digital Forensics CTF Competition

Date: 5/28/2021

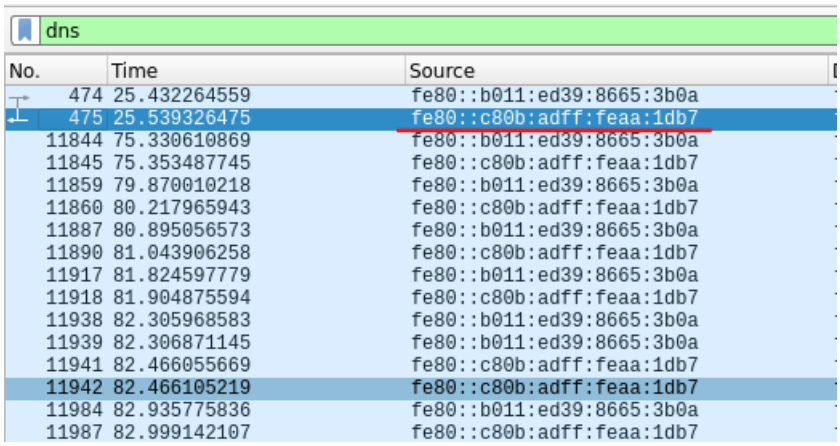
On 5/17/2021 the third week of the competition of the United Nation Office of Drugs and Cybercrime (UNODC) CTF had begun. The third week's challenge was based on network analysis. This was a very interesting challenge to me due to the way the questions were set. Some questions were very trick but doable and some were very complicated which required you to think outside the box. In this week we were faced with 13 questions which were supposed to be answered based on the network dump which was provided.



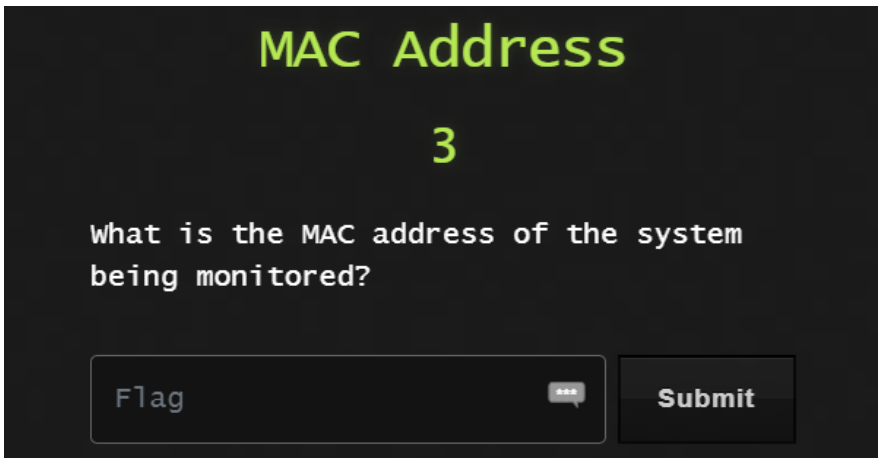
Starting off with the first question which was called DNS we were supposed to find the IPv6 of the provided IP as shown below



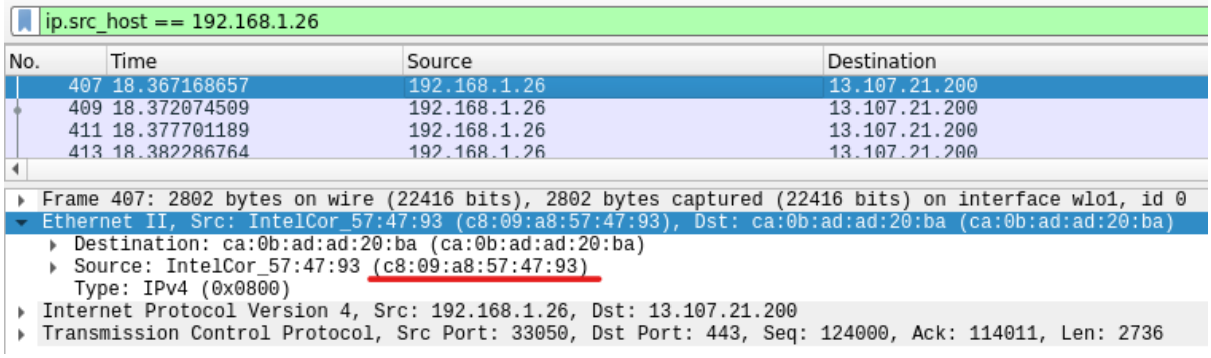
Since the question involved DNS, first of all I had to filter the packets to show only DNS traffic. After analysing the filter result, I had seen the IPv6 of the IP. We would actually see that 192.168.1.26 was making a DNS request to the 192.168.1.10, meaning that we were supposed to get the IPv6 of the 192.168.1.10 IP which shown below.



Moving to the next challenge, this one was called Mac address where in this task were asked to check the MAC address of the system being monitored.

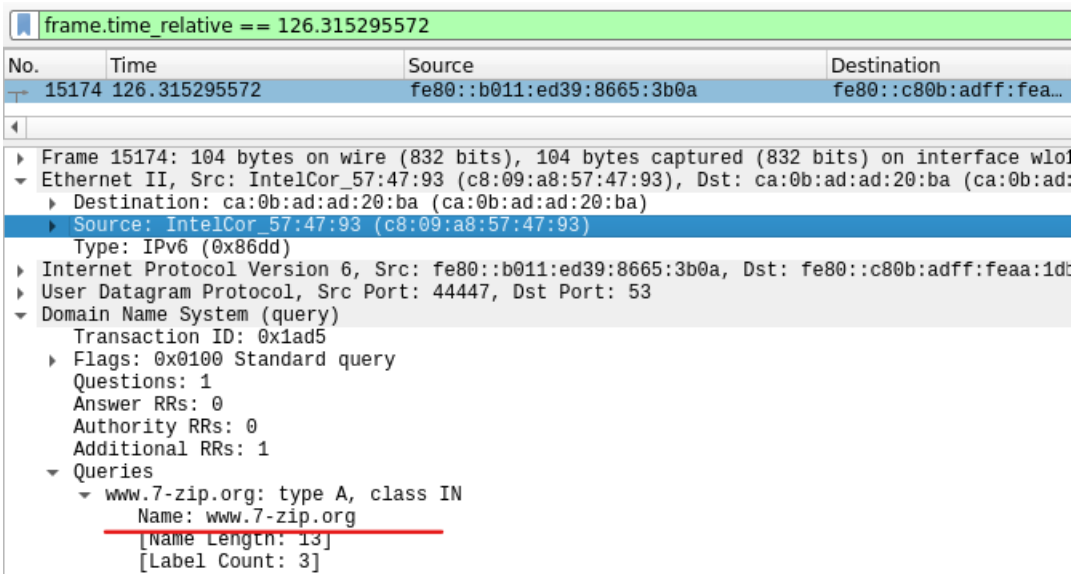


Here we need to find the IP address that appears the most is most likely the one doing the monitoring in the network which in our case it's the 192.168.1.26 ip.

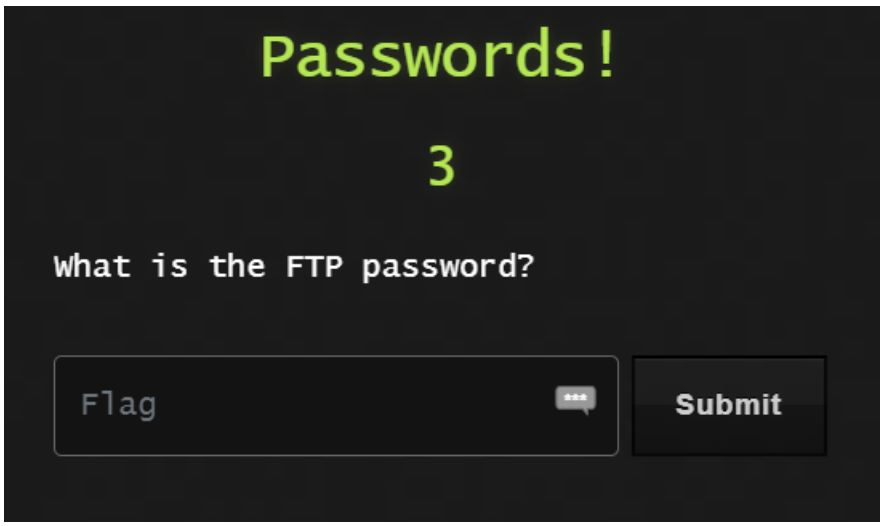


The next question was called packets where we were given a packet number to look for the domain that the user was looking up for.

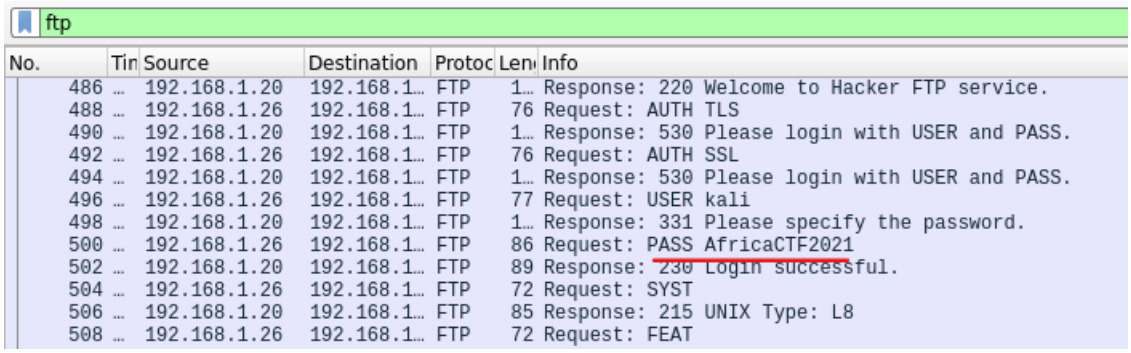
Going to the exact packet under the domain name system section we could find the domain that the user was querying on that frame as show below.



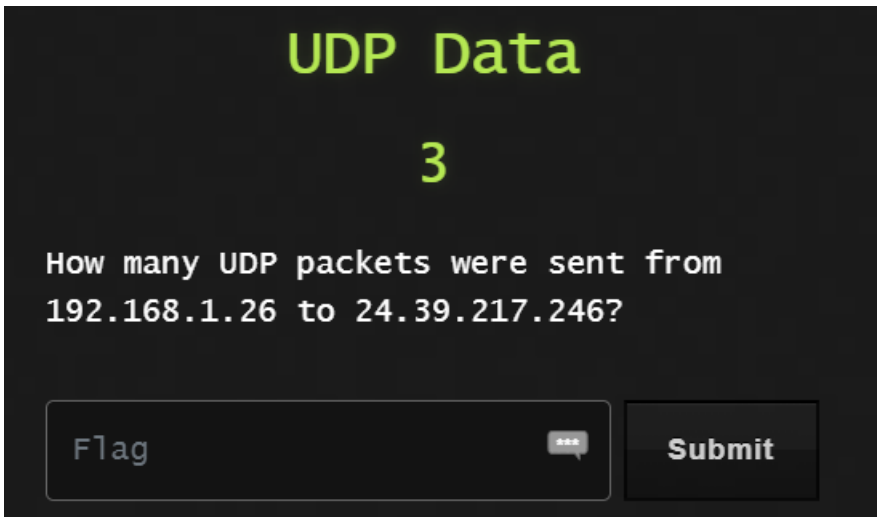
The next question was named passwords where we asked to look for the password of the FTP which was used to login in.



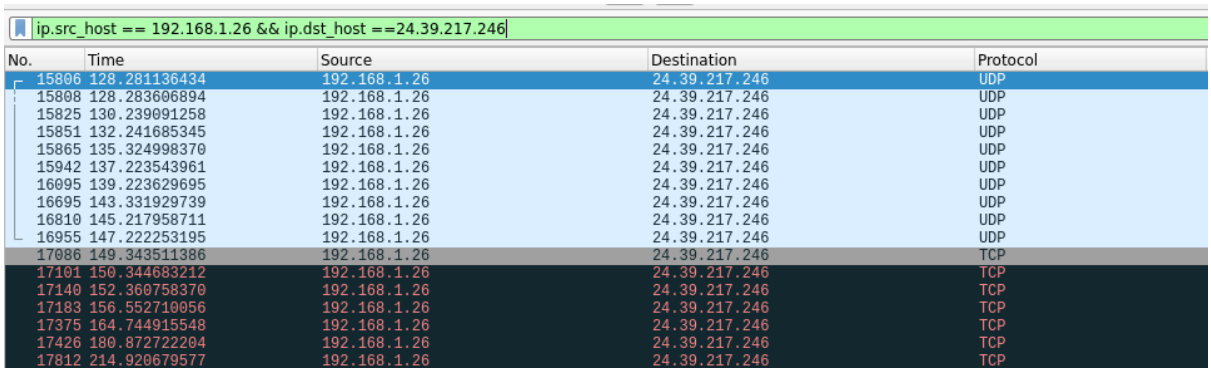
FTP are considered to be insecure source of protocol because it usually sends traffic in plain text, having that in mind I had filtered all the network protocols to display only ftp traffics and as shown below I had found the password the user had entered.



The next challenge was to identify how many packets of UDP were sent from the host 192.168.1.26 to the destination 24.39.217.246.



This was accomplished by using a filter where I specified the specific source and destination on the filter bar which brought me all packets associated with the parameter specified in the filter.

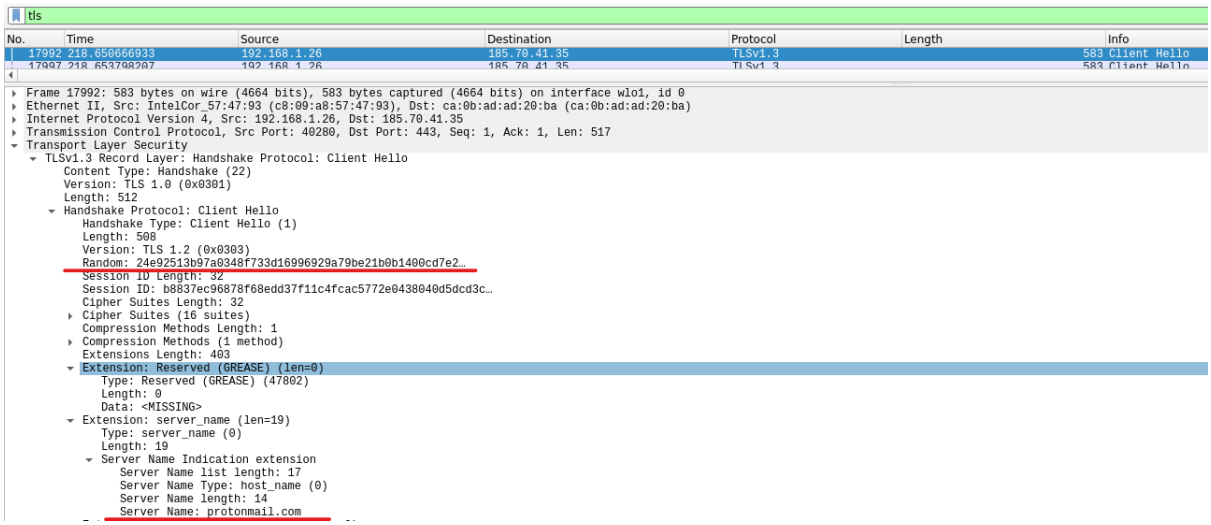


Going further to the next question it was called domains where we were supposed to find the first TLS 1.3 client random that was used to establish a connection to the proton mail.

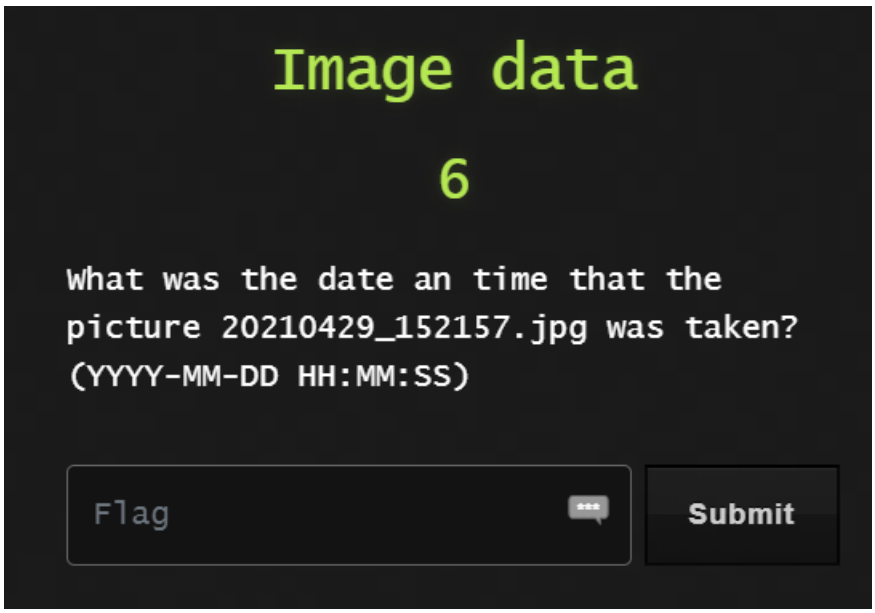
client random is whenever a TLS connection is being setup the client sends in a random session information and the server starts the encryption process and sends some information back and then they negotiate and then make a secure connection.

I filtered the packets to display all TLS packets on the network and went to the specific version I wanted which was version 1.3.

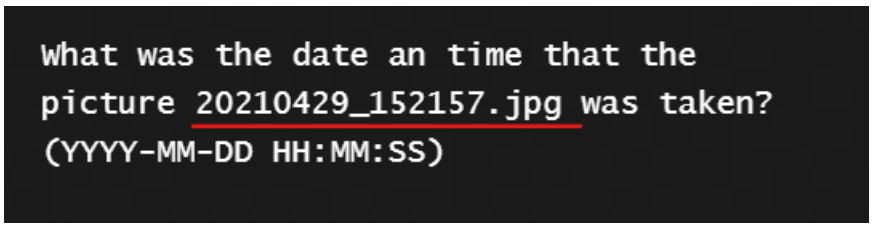
When you scroll down to TLS number 17992 and look at the Transport layer security section you will find the random number of protonmail.com as shown below.



The next challenge was a very straight forward challenge which required a little bit of thinking. This challenge requested for the date and time of the picture 20210429_152157.jpg was taken.

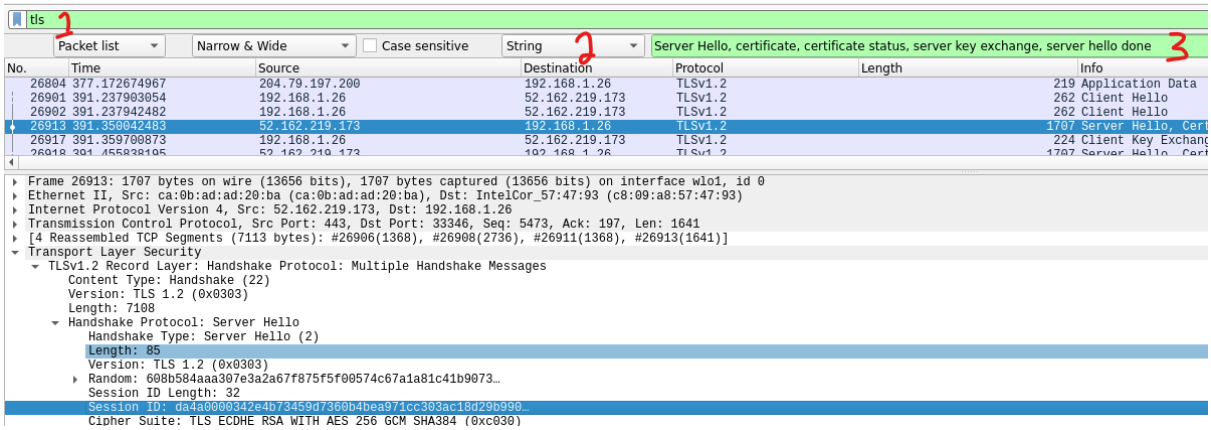


I would say this was a tricky question because the answer to this question was right in front of my eyes, the image name was actually the answer to this question.



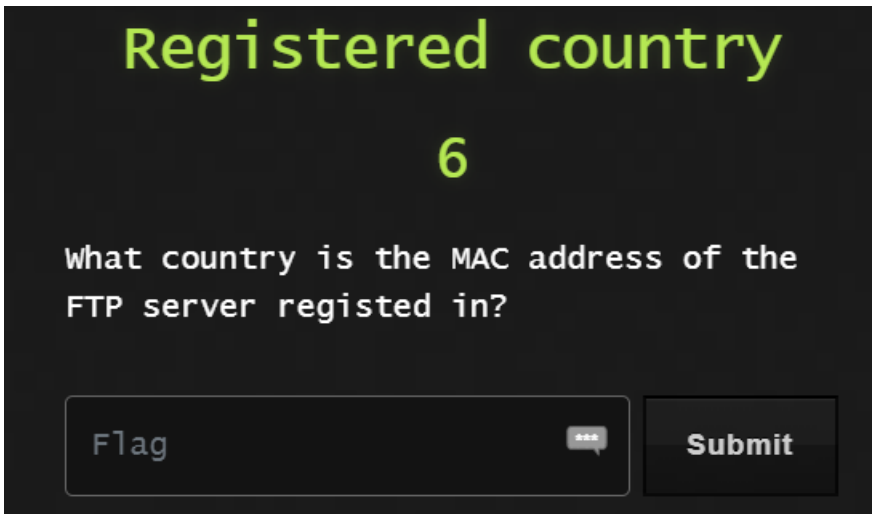
Going to the next question which was finding the server certificate public key that was used on the TLS session specified in the image below

First of all you had to filter the traffic to display on TLS packets, then press control + F to find then change to strings and then search for **Server Hello, certificate, certificate status, server key exchange, server hello done** once done scroll down and check on the transport layer security (TLS) section and check for the session ID if it matches the one provided in the question.

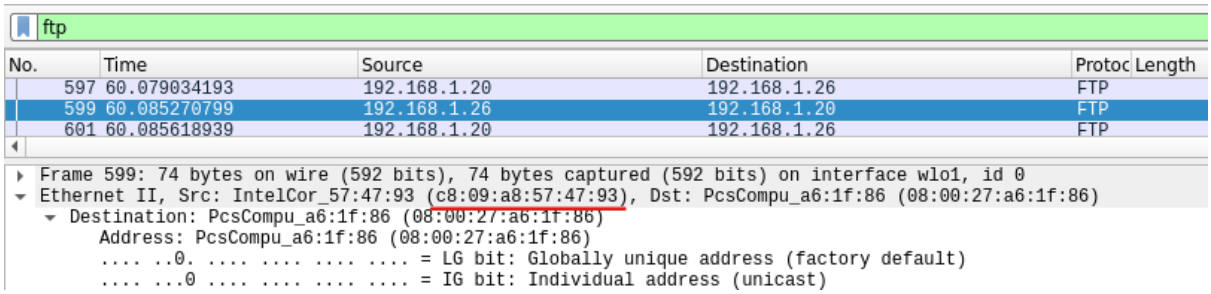


When you find the packet under TLS scroll down till you find a section that says handshake protocol: Server key Exchange when you expand it you will find the public key

This was an interesting question which I must say I really enjoyed doing, the question was what country the MAC address of the FTP server registered in. is



First of all, I began by filtering the traffic to display only FTP traffic, then we had to identify which of the IPv4 was the ftp and in our case the client was 192.168.1.29 and our FTP server was 192.168.1.20. looking at its information on the ethernet section we could see that the mac address of the FTP server is as shown below.



Next was to go in cool and search for a website which can identify where the mac address came from, I got information about the company address and after checking the address on google map.

c8:09:a8:57:47:93

Enter any MAC Address or OUI to check its vendor or enter a vendor name to check its MAC Address ranges and details.

Search

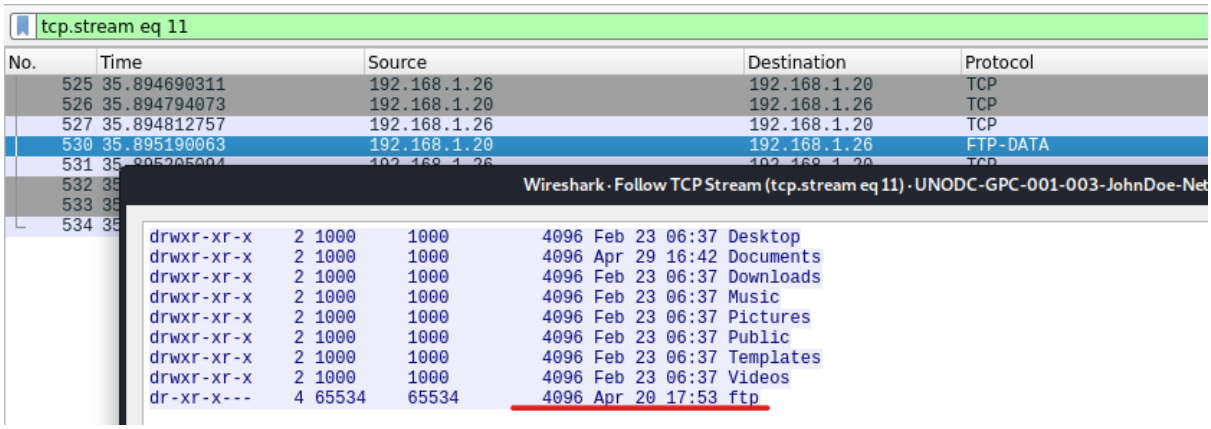
Result for: C8:09:A8:57:47:93

Address Prefix	C8:09:A8
Vendor / Company	Intel Corporate
Start Address	C809A8000000
End Address	C809A8FFFFFF
Company Address	Lot 8, Jalan Hi-Tech 2/3 Kulim Kedah 09000 My

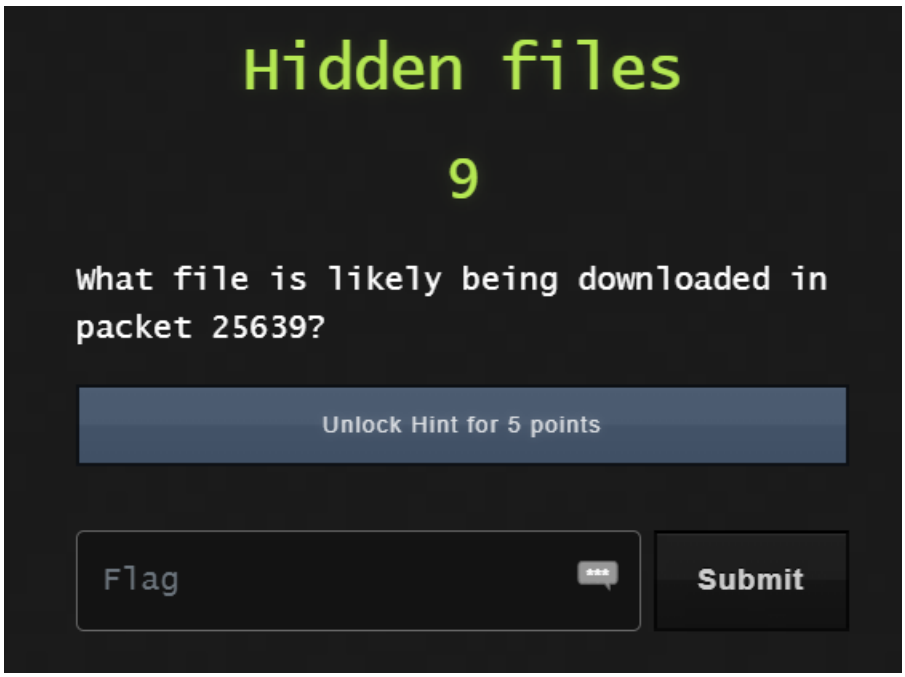
Google map identified that the company was located in Malaysia as shown in the below image

The next challenge was named suspicious folder where we were supposed to find the non-standard folder which was created in the ftp and look for its date and time the folder was created

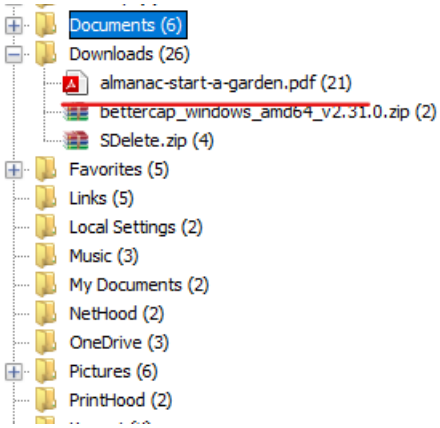
The first thing I had to filter the data for ftp-data, since all the traffic of which contained ftp-data was filtered in then I followed the tcp stream and saw the non-standard folder which was called ftp.



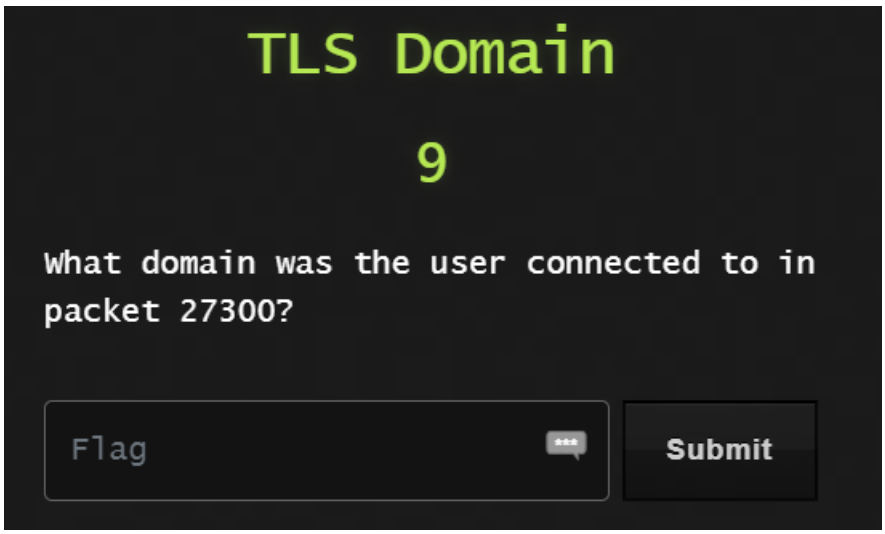
The next challenge was named hidden files which required you to find out what file on packet number 25639 was being downloaded.



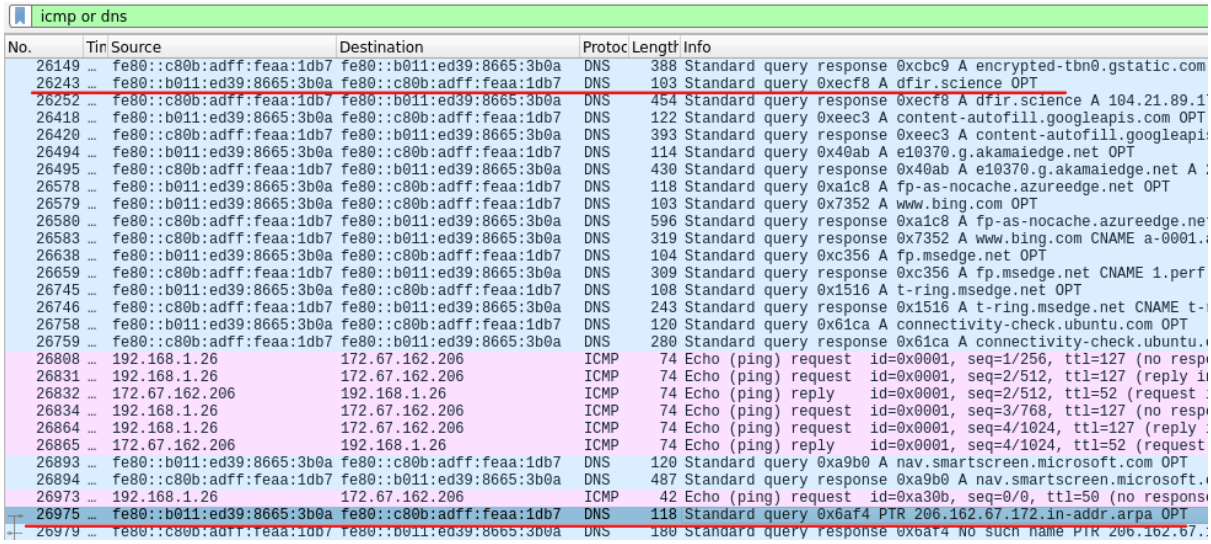
Going to the image file and going to download we could see that the file that was being downloaded was almanac-start-a-garden



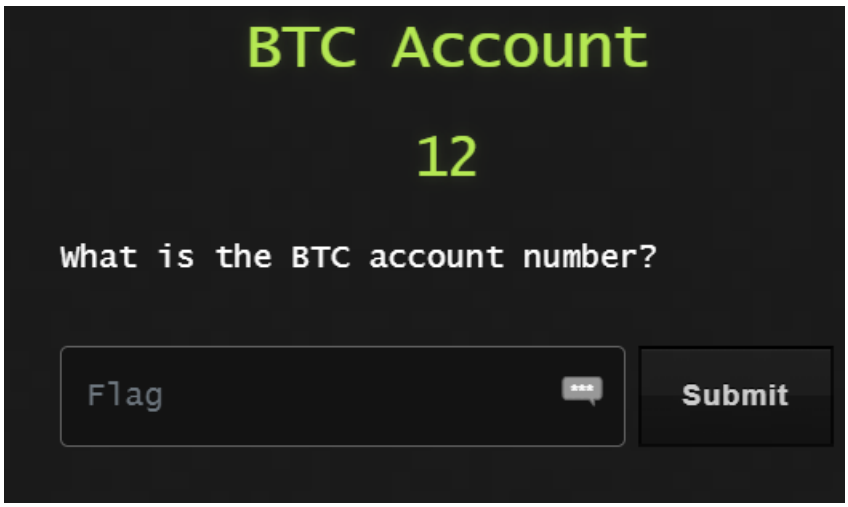
The next question was named TLS Domain, where we were required to find out what the user in was connected to in packet 27300



I filtered the traffic to DNS and after examining the traffic I realized that the user was connected to the domain dfir.science as shown below



This was the final question which was the most complicated of them all which also in the process I managed to learn more about hashcat. The question required us to look for the BTC account number, initially I did not know what BTC was but google it up I realized it meant bitcoin account.

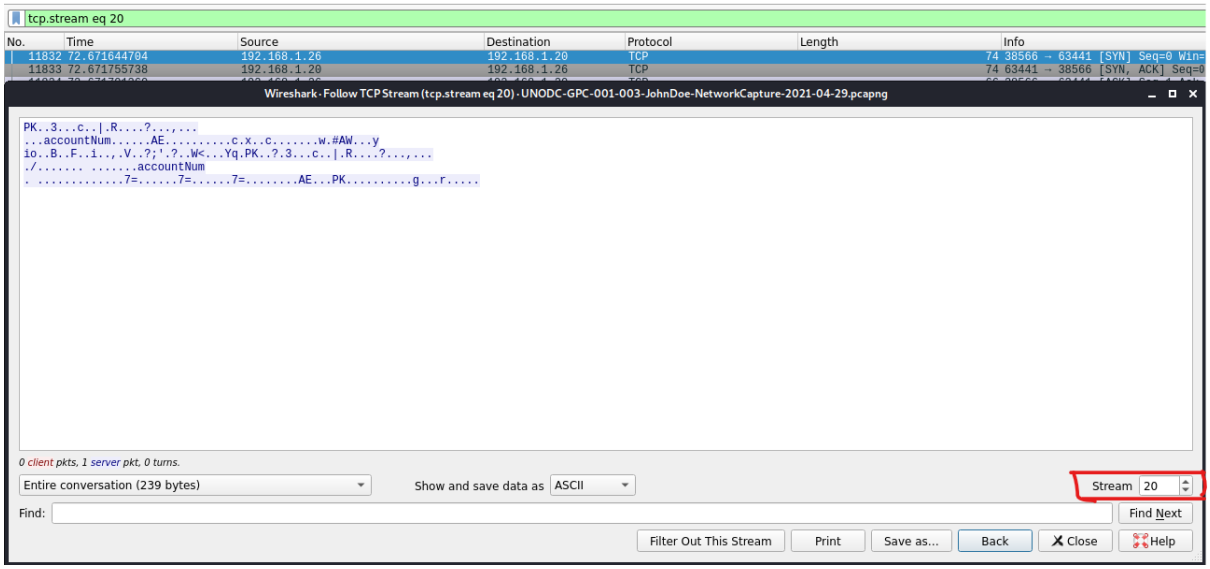


The answer lies in the FTP packets. Going through the FTP traffic you will realize that a file named AccountNum.zip was downloaded from the server, but after investigating I found out the file was permanently deleted from the system.

```
CWD Documents
250 Directory successfully changed.
PWD
257 "/home/kali/Documents" is the current directory
PASV
227 Entering Passive Mode (192,168,1,20,130,4).
LIST
150 Here comes the directory listing.
226 Directory send OK.
MDTM accountNum.zip
213 20210429164247
```

The remaining way to get the file was to extract it from wireshark.

In the FTP you need to right click on one packet then follow TCP stream, going through the streams it was stream 20 which had the zipped file where we extracted it and saved it with a **.zip** extension



After successfully you needed to extract the hash of the zipped file using zip2john. **zip2john AccountNum.zip > AccountNum.hashes**

Once the hash has been extracted it was time to crack the hash of the zip file and get the password.

We had to create a custom rule file which was run side by side with rockyou in order to get the answer. What I did was to take the 3 previous passwords which was AfricaCTF2021, AFR1CA! and ctf2021 as the main words for the rule set. Then I had broken them further down for example AfricaCTF2021 I would break it down to Africa, ctf and 2021 etc.

In order to run the rule list with rockyou I had to append the list. Before getting into it we need to understand how custom rules are created in hashcat.

Let take for example the word cat, you can manually create random combinations which can be possible passwords example c@t or c4t etc, now instead of doing that manually you can use hashcat to create that for you using rules example if you want to capitalize the first letter you would use the rule $c to capitalize all you will use the command $u etc.

Hashcat has the ability to append and prepend characters. Append means that all character would be sent to the right and to achieve that you would put $X where X is a character you want to append and if you want to prepend it means that all the characters would be sent to the left but there is a catch with appending where the user must be very careful.

Example assuming your password is kitcat but on your wordlists there is only cat, normally you would add the word kit before cat but with hashcat you can prepend the process by doing this in every character ^k^i^t.

Now here is the logic if you want to have the word kitcat and your wordlist has cat only and you want to add kit in your wordlist before cat you cant prepend the word ^k^i^t cause if you do it hashcat would insert the word kit into cat as tikcat which is not the password we want but if you want it to be kitcat you must reverse the character you want to append which in our example it would be ^t^i^k instead of ^k^i^t and with this hashcat will inset it as kit before cat. So basically, what hashcat does in prepending it takes the first letter on the right and inserts it as the first letter in your wordlist.

As you can see on our custom dictionary rule set for the first word, we reversed the word Africactf2021 to ^1^2^0^1^f^t^c^a^c^i^r^f^A so that when this rule is run in a wordlist for example dead then it would read as Africactf2021dead during computation.

```
└─# cat custom-dic
^1^2^0^2^f^t^c^a^c^i^r^f^A
^1^2^0^2^f^t^c
^A^C^1^R^F^A^!
^A^C^1^R^F^A^
^a^c^i^r^f^A
^F^T^C
^f^t^c
^1^2^0^2
^a^c^1^r^f^A
^a^c^i^r^f^a
^a^c^1^r^f^a
```

Once the custom rule was ready, I started the process of cracking the hash.

```
└─# hashcat -m 13600 -a 0 AccountNum.hashes /usr/share/wordlists/rockyou.txt -r custom-dic
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.5, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG
=============================================================================================
* Device #1: pthread-Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 1429/1493 MB (512 MB allocat

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Skipping invalid or unsupported rule in file custom-dic on line 4: ^A^C^1^R^F^A^
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 10
```

One hour later the password was extracted as shown below

```
Candidates.#1....: africafelton -> africapetey

$zip2$*0*1*0*ee15e8c7b119a263*b778*2b*1cc963b2a5bee8ff04df77f9234157dfdb0f790a696fe
1d71c5971c1*$/zip2$:CTFPass!
```

Running the password into the zipped file and I got the bitcoin account