ARYAMAN MISHRA

19BCE1027

EXERCISE 4

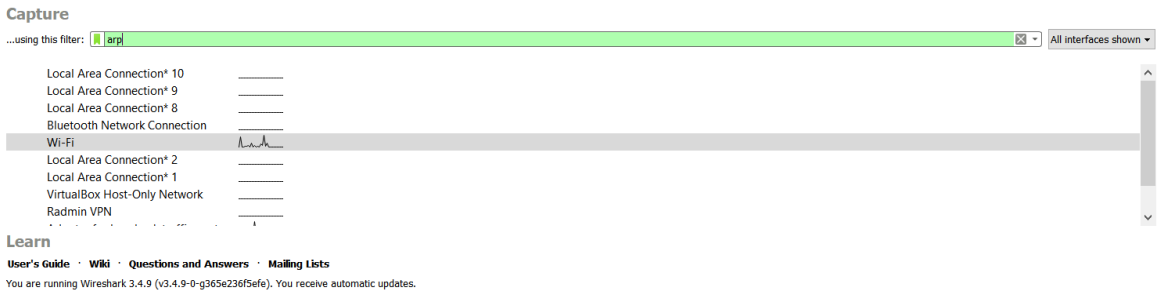
AIM-Packet Sniffing Using Wireshark software.

```
C:\WINDOWS\system32>arp -a

Interface: 169.254.85.211 --- 0x6
  Internet Address      Physical Address      Type
  224.0.0.22            01-00-5e-00-00-16     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 26.99.13.224 --- 0xd
  Internet Address      Physical Address      Type
  224.0.0.22            01-00-5e-00-00-16     static

Interface: 192.168.29.120 --- 0x11
  Internet Address      Physical Address      Type
  192.168.29.1          30-49-50-2e-aa-33     dynamic
  224.0.0.22            01-00-5e-00-00-16     static
```

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::6053:a654:6606:5f99%17
   IPv4 Address. . . . . . . . . . . : 192.168.29.120
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.29.1
```

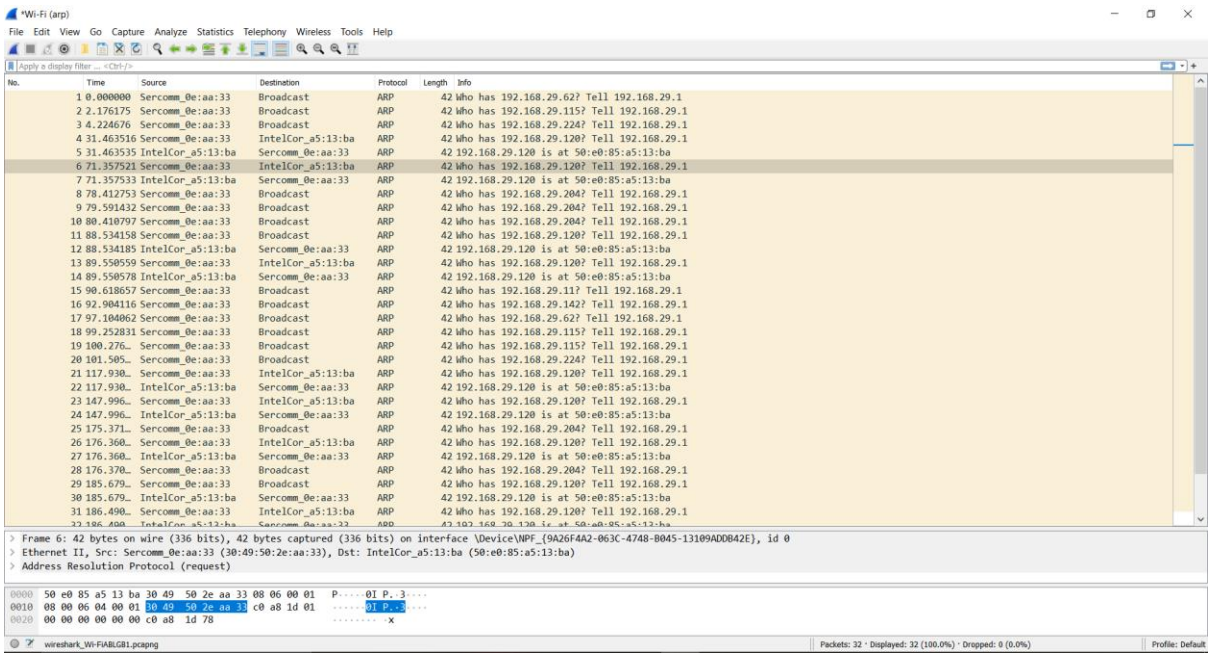
My IP address throughout the document : 192.168.29.120

1. Protocol Analysis - Address Resolution Protocol (ARP)
   1)Open wireshark and start capturing the packets.



   2) Then open the browser and open any website.

3) Go to the wire shark and apply filter “arp”. It will show all the packets using ARP protocol.

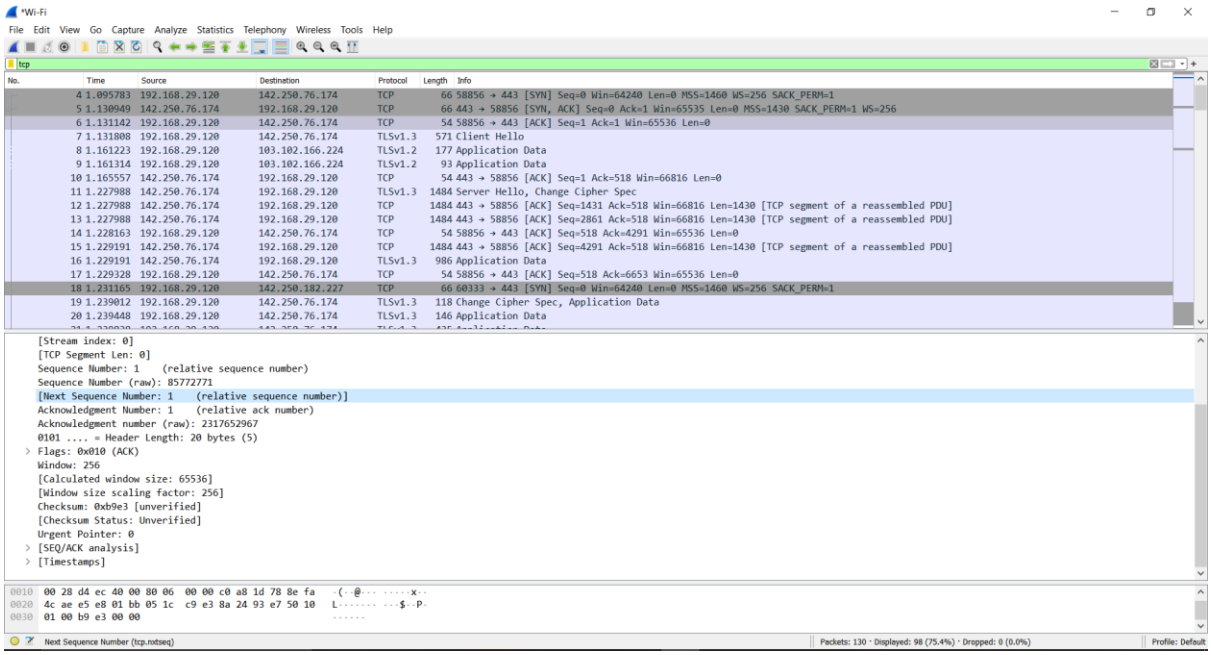


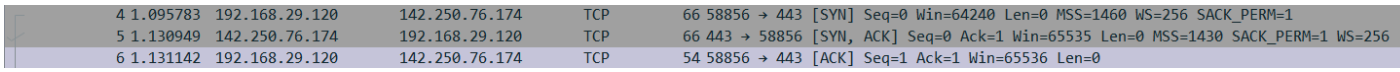
2. TCP 3-way handshake

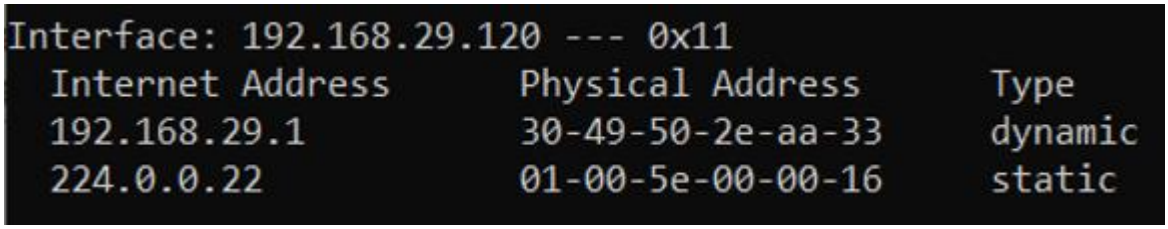
Procedure of 3 way handshake:

It is the process used to establish the communication between client and server. It is 3 steps process:

1) Client will establish the connection with the server by sending the synchronized sequence number (SYN) which will help to inform the server that the client is going to start the communication with the sequence number.
2) The server will respond to the request sends (SYN,ACK) which indicates the request is received
3) Client acknowledges the response of the server After these steps they can start communicating with each other.







1. Go to the browser and open any URL.

2. Start the wireshark capture.

3. Go to the web page and refresh.

4. Filter the packets by command “tcp”.

3. Password Cracking

a. http site

For extracting the password from the website the site must be using http protocol for https protocol we can't use this method. For Ex. VIT's Vproprl portal is using the http protocol so, we will try to extract password form that website. Steps involved are:

1) Open the wireshark and start capturing the packets.

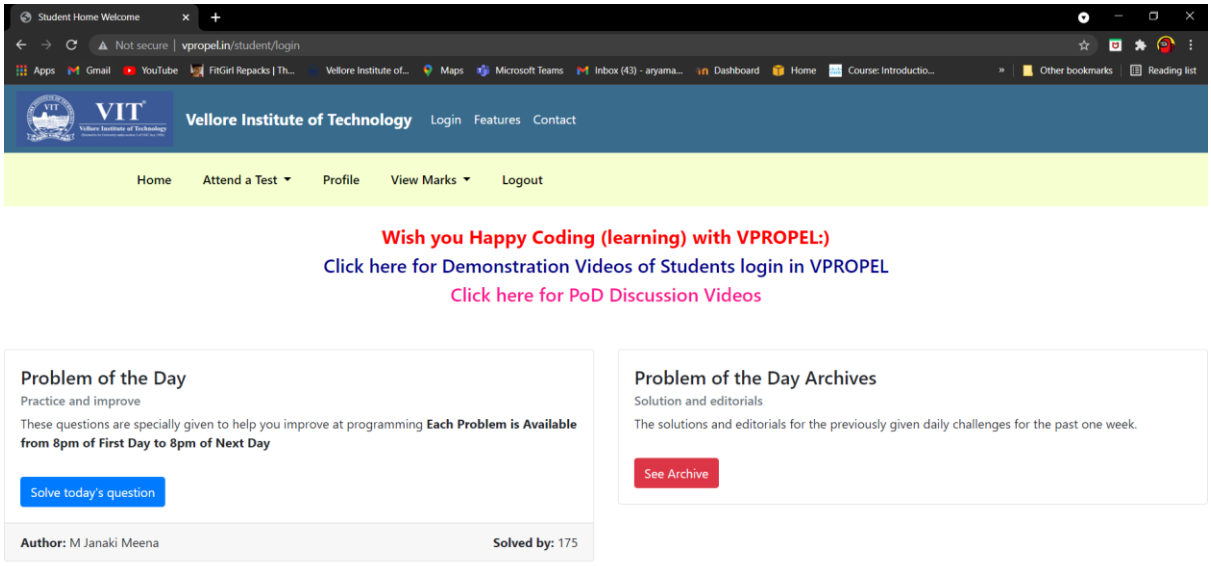
2) Open the web browser and go to vpropel portal.

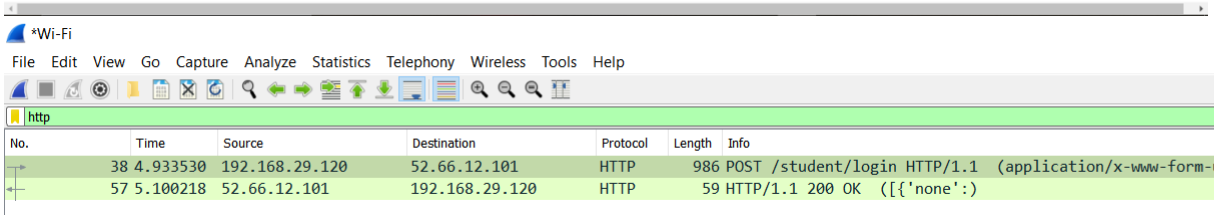
3) Login through your credentials.

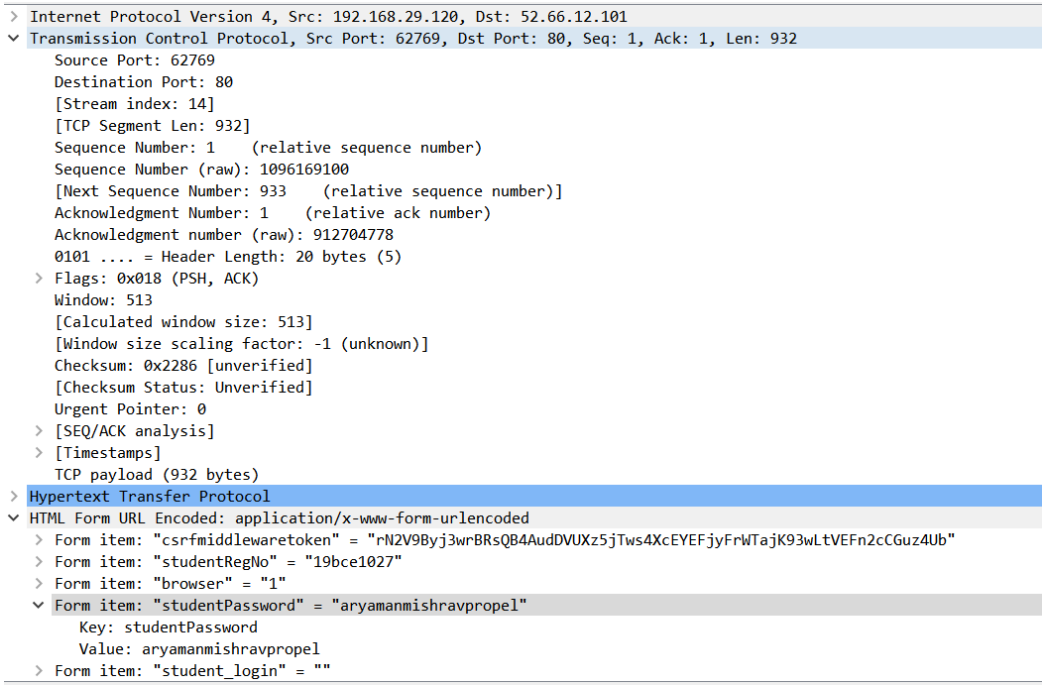
4) Go to wireshark and filter out http protocol packets.

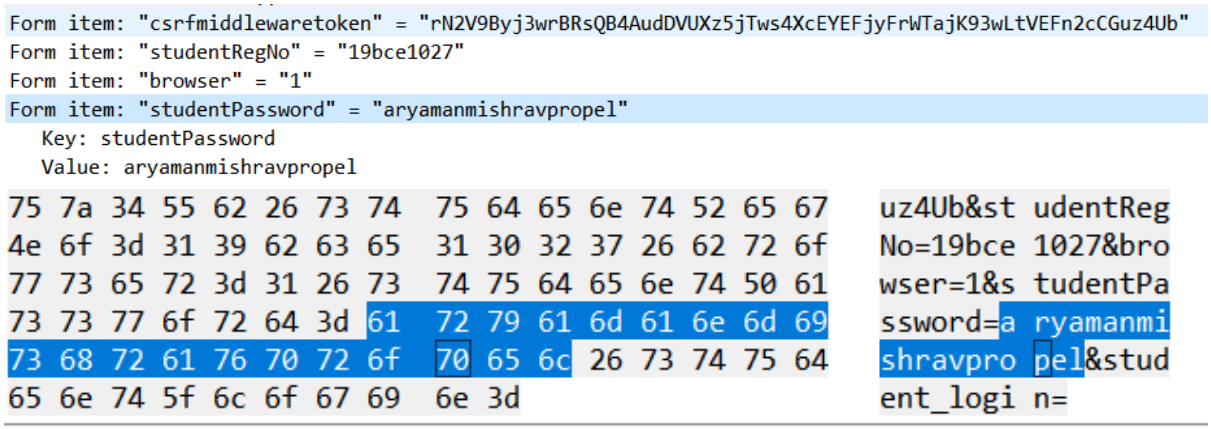
5) Find out the POST type PACKET and go to the analysis section.

6) There we can find StudentLogin info.









b.FTP Server

Initial Setup

Before we begin, let's run a simple Nmap scan on our target to make sure the FTP service is present. We will be using Metasploitable 2 as the target and Kali Linux as the attacking machine.

~# nmap -sV 10.10.0.50 -p 21

Starting Nmap 7.80 (https://nmap.org) at 2020-03-10 11:10 CDT

Nmap scan report for 10.10.0.50

Host is up (0.00067s latency).

```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
MAC Address: 00:1D:09:55:B1:3B (Dell)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.82 seconds
```

Great, it looks like it's up and open.

Next, let's create two text files, one for usernames and one for passwords. In a real engagement, we'd want to use files with much larger data sets, but for demonstration purposes, we'll keep these short to speed up the whole process.

Using your favorite text editor, create a file, and add a few common usernames:

```
root
admin
user
ftp
steve
```

And do the same thing for the passwords:

```
password
s3cr3t
user
Password1
hunter2
```

Now we should be good to go.

Ncrack

The first tool we'll look at today is Ncrack. Simply type **ncrack** in the terminal to display the usage information and available options:

```
~# ncrack

Ncrack 0.7 ( http://ncrack.org )
Usage: ncrack [Options] {target and service specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iX <inputfilename>: Input from Nmap's -oX XML output format
  -iN <inputfilename>: Input from Nmap's -oN Normal output format
  -iL <inputfilename>: Input from list of hosts/networks
```

--exclude <host1[,host2][,host3],...>: Exclude hosts/networks
--excludefile <exclude_file>: Exclude list from file
SERVICE SPECIFICATION:
Can pass target specific services in <service>://target (standard) notation or
using -p which will be applied to all hosts in non-standard notation.
Service arguments can be specified to be host-specific, type of service-specific
(-m) or global (-g). Ex: ssh://10.0.0.10,at=10,cl=30 -m ssh:at=50 -g cd=3000
Ex2: ncrack -p ssh,ftp:3500,25 10.0.0.10 scanme.nmap.org google.com:80,ssl
-p <service-list>: services will be applied to all non-standard notation hosts
-m <service>:<options>: options will be applied to all services of this type
-g <options>: options will be applied to every service globally
Misc options:
ssl: enable SSL over this service
path <name>: used in modules like HTTP ('=' needs escaping if used)
db <name>: used in modules like MongoDB to specify the database
domain <name>: used in modules like WinRM to specify the domain
TIMING AND PERFORMANCE:
Options which take <time> are in seconds, unless you append 'ms'
(milliseconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
Service-specific options:
cl (min connection limit): minimum number of concurrent parallel connections
CL (max connection limit): maximum number of concurrent parallel connections
at (authentication tries): authentication attempts per connection
cd (connection delay): delay <time> between each connection initiation
cr (connection retries): caps number of service connection attempts
to (time-out): maximum cracking <time> for service, regardless of success so far
-T<0-5>: Set timing template (higher is faster)
--connection-limit <number>: threshold for total concurrent connections
--stealthy-linear: try credentials using only one connection against each specified host
until you hit the same host again. Overrides all other timing options.
AUTHENTICATION:
-U <filename>: username file
-P <filename>: password file
--user <username_list>: comma-separated username list
--pass <password_list>: comma-separated password list
--passwords-first: Iterate password list for each username. Default is opposite.
--pairwise: Choose usernames and passwords in pairs.
OUTPUT:
-oN/-oX <file>: Output scan in normal and XML format, respectively, to the given filename.
-oA <basename>: Output in the two major formats at once
-v: Increase verbosity level (use twice or more for greater effect)
-d[level]: Set or increase debugging level (Up to 10 is meaningful)
--nsock-trace <level>: Set nsock trace level (Valid range: 0 - 10)
--log-errors: Log errors/warnings to the normal-format output file
--append-output: Append to rather than clobber specified output files
MISC:
--resume <file>: Continue previously saved session
--save <file>: Save restoration file with specific filename

```
  -f: quit cracking service after one found credential
  -6: Enable IPv6 cracking
  -sL or --list: only list hosts and services
  --datadir <dirname>: Specify custom Ncrack data file location
  --proxy <type://proxy:port>: Make connections via socks4, 4a, http.
  -V: Print version number
  -h: Print this help summary page.
MODULES:
  SSH, RDP, FTP, Telnet, HTTP(S), Wordpress, POP3(S), IMAP, CVS, SMB, VNC, SIP, Redis, PostgreSQL, MQTT,
MySQL, MSSQL, MongoDB, Cassandra, WinRM, OWA, DICOM
EXAMPLES:
  ncrack -v --user root localhost:22
  ncrack -v -T5 https://192.168.0.1
  ncrack -v -iX ~/nmap.xml -g CL=5,to=1h
SEE THE MAN PAGE (http://nmap.org/ncrack/man.html) FOR MORE OPTIONS AND EXAMPLES
```

As you can see, there are a lot of options here, but for now, we'll stick to the basics.

We can use the **-U** flag to set the file containing usernames, and the **-P** flag to set the file containing passwords. Then, specify the service (FTP) followed by the IP address of our target:

```
~# ncrack -U usernames.txt -P passwords.txt ftp://10.10.0.50

Starting Ncrack 0.7 ( http://ncrack.org ) at 2020-03-10 11:24 CDT

Discovered credentials for ftp on 10.10.0.50 21/tcp:
10.10.0.50 21/tcp ftp: 'ftp' 'password'
10.10.0.50 21/tcp ftp: 'ftp' 's3cr3t'
10.10.0.50 21/tcp ftp: 'ftp' 'user'
10.10.0.50 21/tcp ftp: 'ftp' 'Password1'
10.10.0.50 21/tcp ftp: 'user' 'user'
10.10.0.50 21/tcp ftp: 'ftp' 'hunter2'

Ncrack done: 1 service scanned in 15.01 seconds.

Ncrack finished.
```

We can see it discovered credentials for **user** and **ftp**; the multiple hits are because anonymous logins are allowed for that user, making any password a valid password. We can also specify the port number explicitly, which is useful if a service is running on a non-default port. Using the **-v** flag gives us a little more information as well:

```
~# ncrack -U usernames.txt -P passwords.txt 10.10.0.50:21 -v

Starting Ncrack 0.7 ( http://ncrack.org ) at 2020-03-10 11:26 CDT

Discovered credentials on ftp://10.10.0.50:21 'ftp' 'password'
Discovered credentials on ftp://10.10.0.50:21 'ftp' 's3cr3t'
Discovered credentials on ftp://10.10.0.50:21 'ftp' 'user'
```

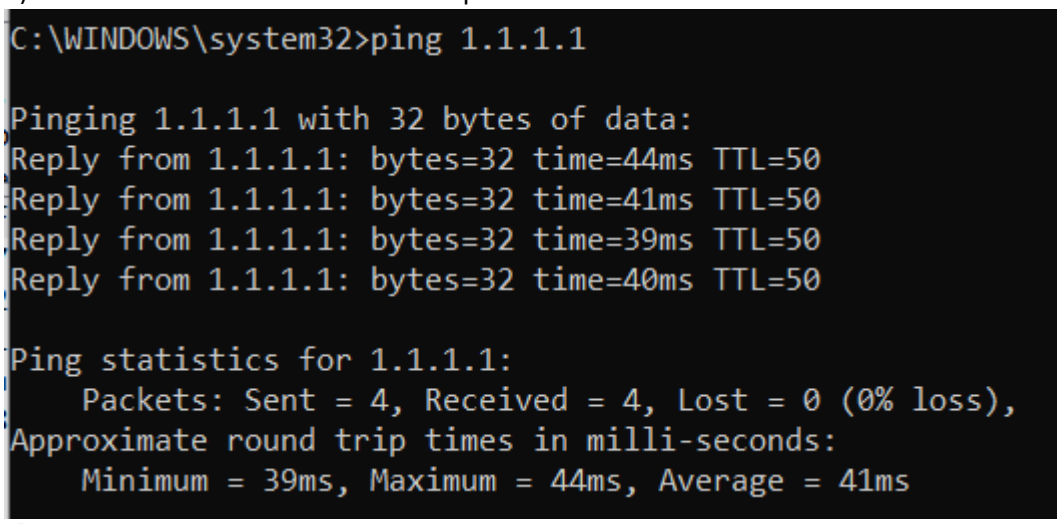
Discovered credentials on ftp://10.10.0.50:21 'user' 'user'
Discovered credentials on ftp://10.10.0.50:21 'ftp' 'Password1'
ftp://10.10.0.50:21 finished.

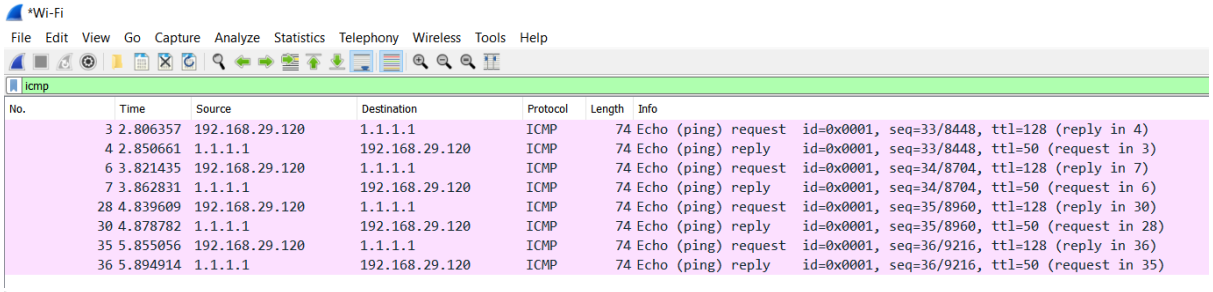
Discovered credentials for ftp on 10.10.0.50 21/tcp:
10.10.0.50 21/tcp ftp: 'ftp' 'password'
10.10.0.50 21/tcp ftp: 'ftp' 's3cr3t'
10.10.0.50 21/tcp ftp: 'ftp' 'user'
10.10.0.50 21/tcp ftp: 'user' 'user'
10.10.0.50 21/tcp ftp: 'ftp' 'Password1'

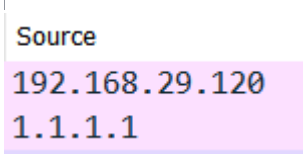
Ncrack done: 1 service scanned in 15.00 seconds.
Probes sent: 17 | timed-out: 0 | prematurely-closed: 0

Ncrack finished.

3. Packet Analysis – Ping packets (ICMP)
   ICMP - Internet control Message Protocol
   We use this to check whether the host or the router is reachable or not in the network Steps Involved are :
   1) Open the wireshark and start capturing the packets.
   2) Go to terminal and type “ping 1.1.1.1”
   3) Go to wireshark and filter the ICMP packets.



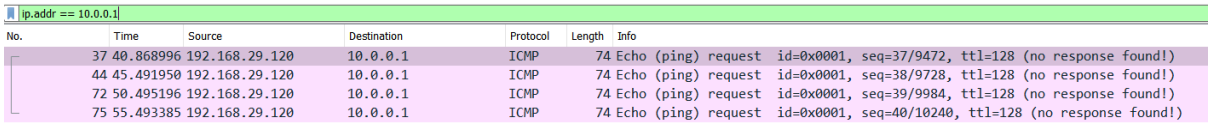




MY IP ADDRESS WILL BE VISIBLE(192.168.29.120)

5. Implement the following filters (Create suitable traffic for each filter, so that you get response for each filter)

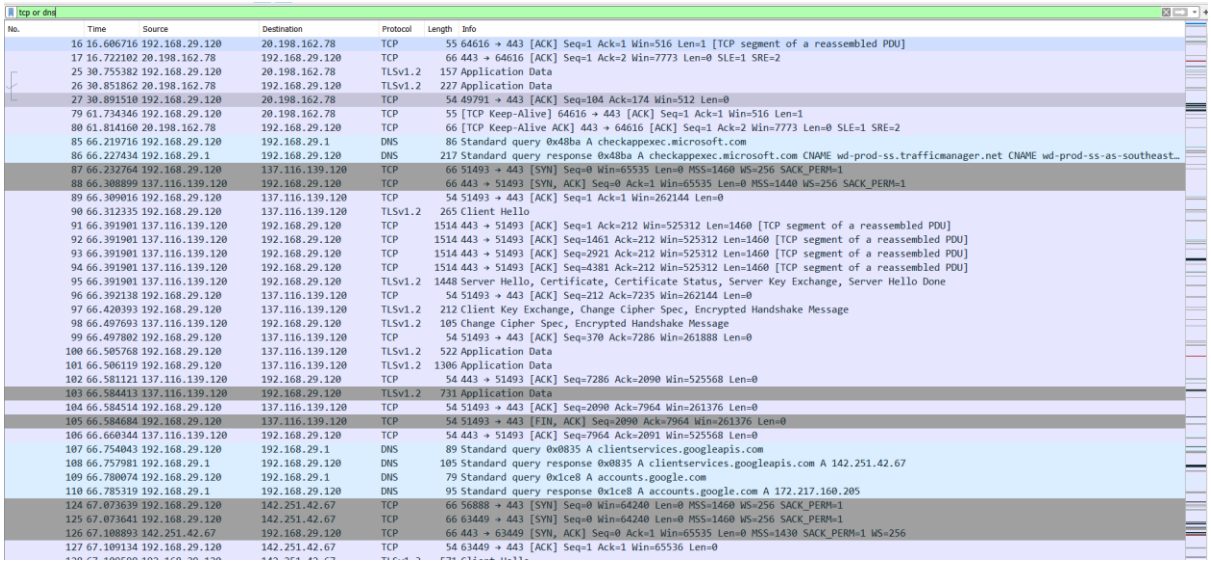
a. ip.addr == 10.0.0.1



ip.addr == 10.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
37	40.868996	192.168.29.120	10.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=37/9472, ttl=128 (no response found!)
44	45.491950	192.168.29.120	10.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=38/9728, ttl=128 (no response found!)
72	50.495196	192.168.29.120	10.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=39/9984, ttl=128 (no response found!)
75	55.493385	192.168.29.120	10.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=40/10240, ttl=128 (no response found!)

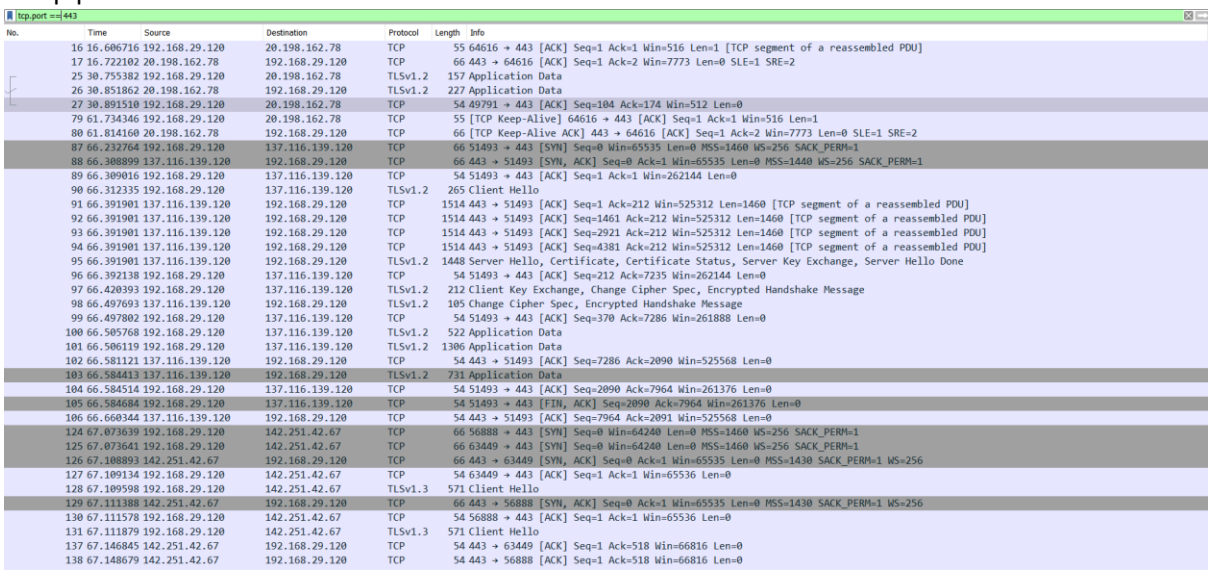
b. tcp or dns



tcp or dns

No.	Time	Source	Destination	Protocol	Length	Info
16	16.606716	192.168.29.120	20.198.162.78	TCP	55	64616 → 443 [ACK] Seq=1 Ack=1 Win=516 Len=1 [TCP segment of a reassembled PDU]
17	16.722102	20.198.162.78	192.168.29.120	TCP	66	443 → 64616 [ACK] Seq=1 Ack=2 Win=7773 Len=0 SLE=1 SRE=2
25	30.755382	192.168.29.120	20.198.162.78	TLSv1.2	157	Application Data
26	30.851862	20.198.162.78	192.168.29.120	TLSv1.2	227	Application Data
27	30.891510	192.168.29.120	20.198.162.78	TCP	54	49791 → 443 [ACK] Seq=104 Ack=174 Win=512 Len=0
79	61.734346	192.168.29.120	20.198.162.78	TCP	55	[TCP Keep-Alive] 64616 → 443 [ACK] Seq=1 Ack=1 Win=516 Len=1
80	61.814160	20.198.162.78	192.168.29.120	TCP	66	[TCP Keep-Alive ACK] 443 → 64616 [ACK] Seq=1 Ack=2 Win=7773 Len=0 SLE=1 SRE=2
85	66.219716	192.168.29.120	192.168.29.1	DNS	86	Standard query 0x48ba A checkappexec.microsoft.com
86	66.227434	192.168.29.1	192.168.29.120	DNS	217	Standard query response 0x48ba A checkappexec.microsoft.com CNAME wd-prod-ss.trafficmanager.net CNAME wd-prod-ss-as-southeast…
87	66.232764	192.168.29.120	137.116.139.120	TCP	66	51493 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
88	66.308899	137.116.139.120	192.168.29.120	TCP	66	443 → 51493 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
89	66.309016	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
90	66.312335	192.168.29.120	137.116.139.120	TLSv1.2	265	Client Hello
91	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=1 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
92	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=1461 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
93	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=2921 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
94	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=4381 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
95	66.391901	137.116.139.120	192.168.29.120	TLSv1.2	1448	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
96	66.392138	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=212 Ack=7235 Win=262144 Len=0
97	66.420393	192.168.29.120	137.116.139.120	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
98	66.497693	137.116.139.120	192.168.29.120	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
99	66.497802	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=370 Ack=7286 Win=261888 Len=0
100	66.505768	192.168.29.120	137.116.139.120	TLSv1.2	522	Application Data
101	66.506119	192.168.29.120	137.116.139.120	TLSv1.2	1306	Application Data
102	66.581121	137.116.139.120	192.168.29.120	TCP	54	443 → 51493 [ACK] Seq=7286 Ack=2090 Win=525568 Len=0
103	66.584413	137.116.139.120	192.168.29.120	TLSv1.2	731	Application Data
104	66.584514	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=2090 Ack=7964 Win=261376 Len=0
105	66.584684	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [FIN, ACK] Seq=2090 Ack=7964 Win=261376 Len=0
106	66.660344	137.116.139.120	192.168.29.120	TCP	54	443 → 51493 [ACK] Seq=7964 Ack=2091 Win=525568 Len=0
107	66.754043	192.168.29.120	192.168.29.1	DNS	89	Standard query 0x0835 A clientservices.googleapis.com
108	66.757981	192.168.29.1	192.168.29.120	DNS	105	Standard query response 0x0835 A clientservices.googleapis.com A 142.251.42.67
109	66.780074	192.168.29.120	192.168.29.1	DNS	79	Standard query 0x1ce8 A accounts.google.com
110	66.785319	192.168.29.1	192.168.29.120	DNS	95	Standard query response 0x1ce8 A accounts.google.com A 172.217.160.205
124	67.073639	192.168.29.120	142.251.42.67	TCP	66	56888 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
125	67.073641	192.168.29.120	142.251.42.67	TCP	66	63449 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
126	67.108893	142.251.42.67	192.168.29.120	TCP	66	443 → 63449 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
127	67.109134	192.168.29.120	142.251.42.67	TCP	54	63449 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0

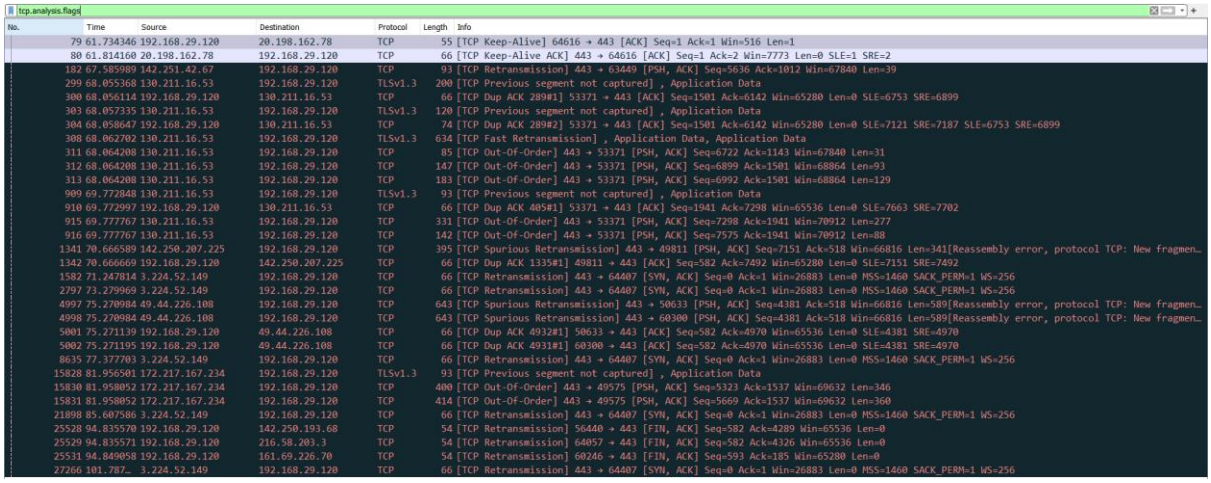
c. tcp.port == 443



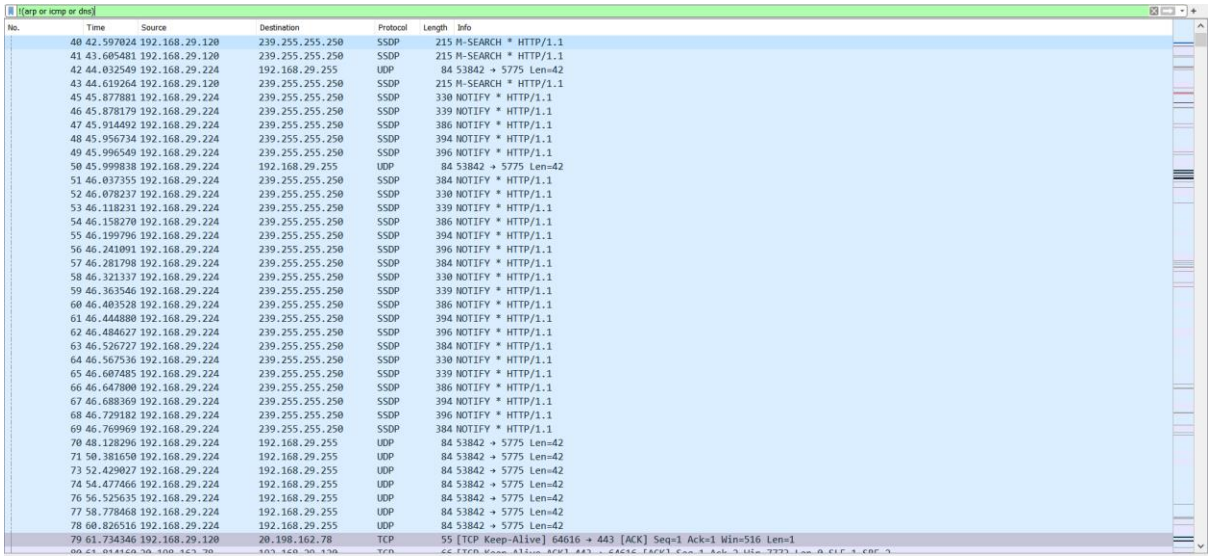
tcp.port == 443

No.	Time	Source	Destination	Protocol	Length	Info
16	16.606716	192.168.29.120	20.198.162.78	TCP	55	64616 → 443 [ACK] Seq=1 Ack=1 Win=516 Len=1 [TCP segment of a reassembled PDU]
17	16.722102	20.198.162.78	192.168.29.120	TCP	66	443 → 64616 [ACK] Seq=1 Ack=2 Win=7773 Len=0 SLE=1 SRE=2
25	30.755382	192.168.29.120	20.198.162.78	TLSv1.2	157	Application Data
26	30.851862	20.198.162.78	192.168.29.120	TLSv1.2	227	Application Data
27	30.891510	192.168.29.120	20.198.162.78	TCP	54	49791 → 443 [ACK] Seq=104 Ack=174 Win=512 Len=0
79	61.734346	192.168.29.120	20.198.162.78	TCP	55	[TCP Keep-Alive] 64616 → 443 [ACK] Seq=1 Ack=1 Win=516 Len=1
80	61.814160	20.198.162.78	192.168.29.120	TCP	66	[TCP Keep-Alive ACK] 443 → 64616 [ACK] Seq=1 Ack=2 Win=7773 Len=0 SLE=1 SRE=2
87	66.232764	192.168.29.120	137.116.139.120	TCP	66	51493 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
88	66.308899	137.116.139.120	192.168.29.120	TCP	66	443 → 51493 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
89	66.309016	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
90	66.312335	192.168.29.120	137.116.139.120	TLSv1.2	265	Client Hello
91	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=1 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
92	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=1461 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
93	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=2921 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
94	66.391901	137.116.139.120	192.168.29.120	TCP	1514	443 → 51493 [ACK] Seq=4381 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
95	66.391901	137.116.139.120	192.168.29.120	TLSv1.2	1448	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
96	66.392138	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=212 Ack=7235 Win=262144 Len=0
97	66.420393	192.168.29.120	137.116.139.120	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
98	66.497693	137.116.139.120	192.168.29.120	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
99	66.497802	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=370 Ack=7286 Win=261888 Len=0
100	66.505768	192.168.29.120	137.116.139.120	TLSv1.2	522	Application Data
101	66.506119	192.168.29.120	137.116.139.120	TLSv1.2	1306	Application Data
102	66.581121	137.116.139.120	192.168.29.120	TCP	54	443 → 51493 [ACK] Seq=7286 Ack=2090 Win=525568 Len=0
103	66.584413	137.116.139.120	192.168.29.120	TLSv1.2	731	Application Data
104	66.584514	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [ACK] Seq=2090 Ack=7964 Win=261376 Len=0
105	66.584684	192.168.29.120	137.116.139.120	TCP	54	51493 → 443 [FIN, ACK] Seq=2090 Ack=7964 Win=261376 Len=0
106	66.660344	137.116.139.120	192.168.29.120	TCP	54	443 → 51493 [ACK] Seq=7964 Ack=2091 Win=525568 Len=0
124	67.073639	192.168.29.120	142.251.42.67	TCP	66	56888 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
125	67.073641	192.168.29.120	142.251.42.67	TCP	66	63449 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
126	67.108893	142.251.42.67	192.168.29.120	TCP	66	443 → 63449 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
127	67.109134	192.168.29.120	142.251.42.67	TCP	54	63449 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
128	67.109598	192.168.29.120	142.251.42.67	TLSv1.3	571	Client Hello
129	67.111388	142.251.42.67	192.168.29.120	TCP	66	443 → 56888 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
130	67.111578	192.168.29.120	142.251.42.67	TCP	54	56888 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
131	67.111879	192.168.29.120	142.251.42.67	TLSv1.3	571	Client Hello
137	67.146845	142.251.42.67	192.168.29.120	TCP	54	443 → 63449 [ACK] Seq=1 Ack=518 Win=66816 Len=0
138	67.148679	142.251.42.67	192.168.29.120	TCP	54	443 → 56888 [ACK] Seq=1 Ack=518 Win=66816 Len=0

d. Tcp.analysis.flags



e. !(arp or icmp or dns)



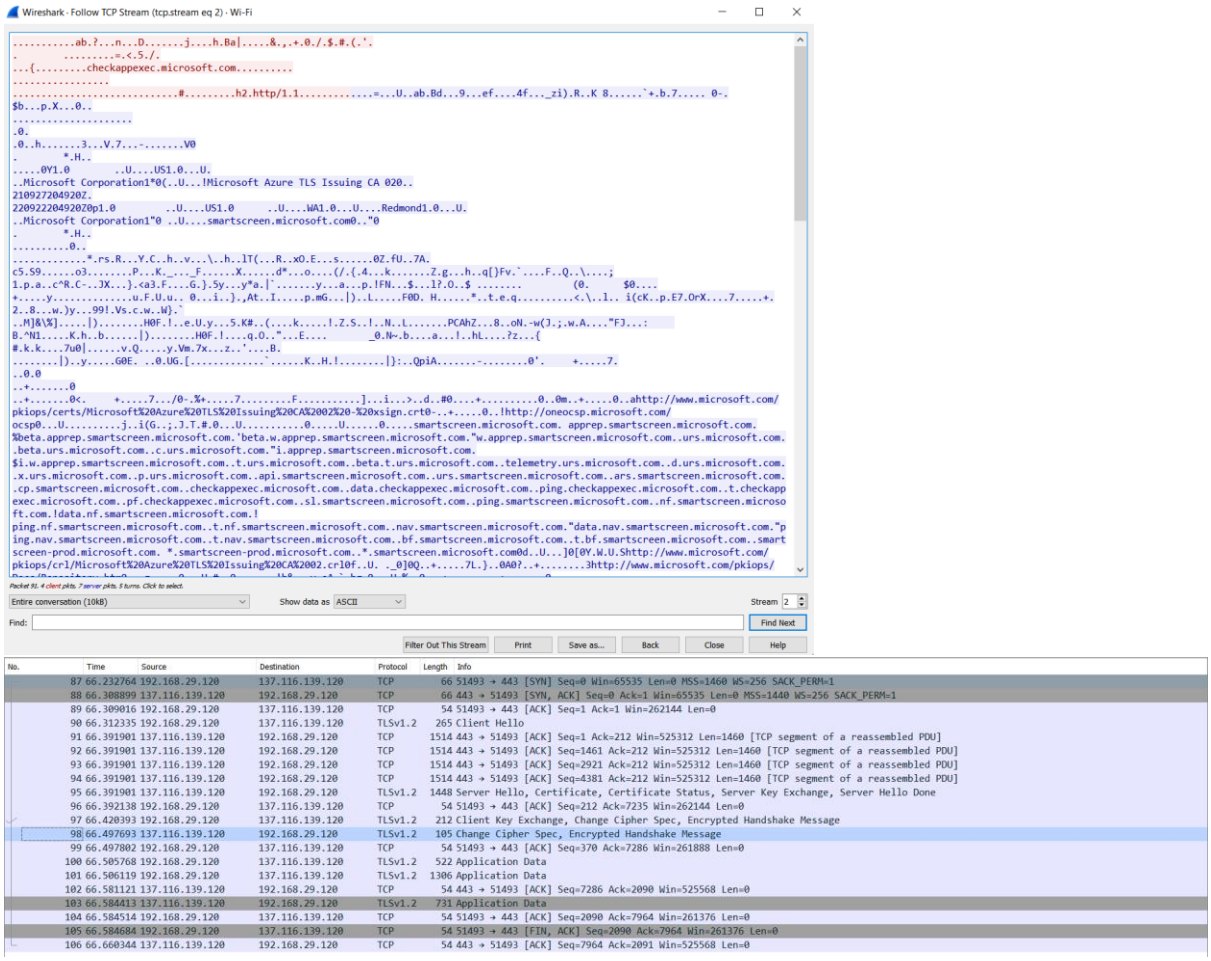
f. follow tcp stream Steps:

1) Open the wireshark and start capturing the packets

2) Select a TCP packet.

3) Go to the analyse section.

4) Then select follow → Follow TCP stream.



g. tcp contains facebook

Steps:

1) Open Browser and go to facebook.com

2) Start capturing in the wireshark

tcp contains facebook

No.	Time	Source	Destination	Protocol	Length	Info
475	10.209628	192.168.29.120	157.240.198.35	TLSv1.3	571	Client Hello
478	10.211099	192.168.29.120	157.240.198.35	TLSv1.3	571	Client Hello
3411	13.899093	192.168.29.120	157.240.198.17	TLSv1.3	571	Client Hello
4232	14.340264	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
4696	14.769602	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
4807	15.056400	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
4990	15.336527	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
5176	15.615058	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
5494	15.906914	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
5784	16.214941	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
5861	16.309787	192.168.29.120	157.240.198.17	TLSv1.3	571	Client Hello
6234	16.517556	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
6552	16.849138	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9375	20.671085	192.168.29.120	157.240.198.17	TLSv1.3	571	Client Hello
9482	21.133274	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9586	21.430728	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9666	21.709112	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9776	21.989473	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9847	22.268460	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
9915	22.548001	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
10091	22.829807	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello
10834	23.232095	192.168.29.120	157.240.198.10	TLSv1.3	571	Client Hello

h. http.response.code == 200

The HTTP 200 OK success status response code indicates that the request has succeeded. ... The meaning of a success depends on the HTTP request method: GET : The resource has been fetched and is transmitted in the message body. HEAD : The representation headers are included in the response without any message body

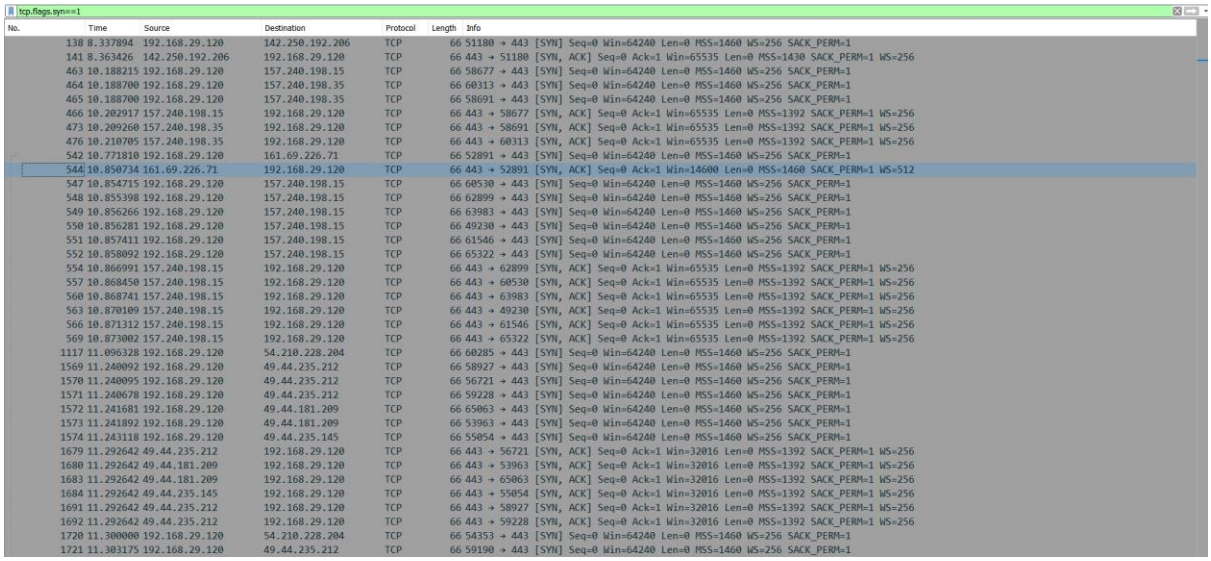
617	12:53:59.666610869	52.66.12.101	192.168.0.111	HTTP	2689	HTTP/1.1 200 OK (text/html)
869	12:54:00.501954382	52.66.12.101	192.168.0.111	HTTP	2692	HTTP/1.1 200 OK (text/html)
1378	12:54:03.495054624	52.66.12.101	192.168.0.111	HTTP	1303	HTTP/1.1 200 OK ([{'false':
11252	13:00:25.094073133	52.66.12.101	192.168.0.111	HTTP	2644	HTTP/1.1 200 OK (text/html)
11733	13:00:53.802641926	52.66.12.101	192.168.0.111	HTTP	962	HTTP/1.1 200 OK ([{'false':
12175	13:01:19.325508010	52.66.12.101	192.168.0.111	HTTP	2976	HTTP/1.1 200 OK (text/html)
12319	13:01:23.695585037	52.66.12.101	192.168.0.111	HTTP	1150	HTTP/1.1 200 OK ([{'false':
12394	13:01:25.956647338	52.66.12.101	192.168.0.111	HTTP	3126	HTTP/1.1 200 OK (text/html)
12536	13:01:32.741699854	52.66.12.101	192.168.0.111	HTTP	2392	HTTP/1.1 200 OK (text/html)
81451	14:07:09.583610680	52.66.12.101	192.168.0.111	HTTP	2690	HTTP/1.1 200 OK (text/html)
81673	14:07:10.764366771	52.66.12.101	192.168.0.111	HTTP	2690	HTTP/1.1 200 OK (text/html)

i. Http.request

http.request

No.	Time	Source	Destination	Protocol	Length	Info
11219	23.566268	192.168.29.120	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
11417	26.579580	192.168.29.120	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1

j. tcp.flags.syn == 1



tcp.flags.syn==1

No.	Time	Source	Destination	Protocol	Length	Info
138	8.337894	192.168.29.120	142.250.192.206	TCP	66	51180 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
141	8.363426	142.250.192.206	192.168.29.120	TCP	66	443 → 51180 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
463	10.188215	192.168.29.120	157.240.198.15	TCP	66	58677 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
464	10.188700	192.168.29.120	157.240.198.35	TCP	66	60313 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
465	10.188700	192.168.29.120	157.240.198.35	TCP	66	58691 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
466	10.202917	157.240.198.15	192.168.29.120	TCP	66	443 → 58677 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
473	10.209260	157.240.198.35	192.168.29.120	TCP	66	443 → 58691 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
476	10.210705	157.240.198.35	192.168.29.120	TCP	66	443 → 60313 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
542	10.771810	192.168.29.120	161.69.226.71	TCP	66	52891 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
544	10.850734	161.69.226.71	192.168.29.120	TCP	66	443 → 52891 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=512
547	10.854715	192.168.29.120	157.240.198.15	TCP	66	60530 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
548	10.855398	192.168.29.120	157.240.198.15	TCP	66	62899 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
549	10.856266	192.168.29.120	157.240.198.15	TCP	66	63983 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
550	10.856281	192.168.29.120	157.240.198.15	TCP	66	49230 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
551	10.857411	192.168.29.120	157.240.198.15	TCP	66	61546 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
552	10.858092	192.168.29.120	157.240.198.15	TCP	66	65322 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
554	10.866991	157.240.198.15	192.168.29.120	TCP	66	443 → 62899 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
557	10.868450	157.240.198.15	192.168.29.120	TCP	66	443 → 60530 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
560	10.868741	157.240.198.15	192.168.29.120	TCP	66	443 → 63983 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
563	10.870109	157.240.198.15	192.168.29.120	TCP	66	443 → 49230 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
566	10.871312	157.240.198.15	192.168.29.120	TCP	66	443 → 61546 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
569	10.873002	157.240.198.15	192.168.29.120	TCP	66	443 → 65322 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1392 SACK_PERM=1 WS=256
1117	11.096328	192.168.29.120	54.210.228.204	TCP	66	60285 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1569	11.240092	192.168.29.120	49.44.235.212	TCP	66	58927 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1570	11.240095	192.168.29.120	49.44.235.212	TCP	66	56721 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1571	11.240678	192.168.29.120	49.44.235.212	TCP	66	59228 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1572	11.241681	192.168.29.120	49.44.181.209	TCP	66	65063 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1573	11.241892	192.168.29.120	49.44.181.209	TCP	66	53963 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1574	11.243118	192.168.29.120	49.44.235.145	TCP	66	55054 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1679	11.292642	49.44.235.212	192.168.29.120	TCP	66	443 → 56721 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1680	11.292642	49.44.181.209	192.168.29.120	TCP	66	443 → 53963 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1683	11.292642	49.44.181.209	192.168.29.120	TCP	66	443 → 65063 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1684	11.292642	49.44.235.145	192.168.29.120	TCP	66	443 → 55054 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1691	11.292642	49.44.235.212	192.168.29.120	TCP	66	443 → 58927 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1692	11.292642	49.44.235.212	192.168.29.120	TCP	66	443 → 59228 [SYN, ACK] Seq=0 Ack=1 Win=32016 Len=0 MSS=1392 SACK_PERM=1 WS=256
1720	11.300000	192.168.29.120	54.210.228.204	TCP	66	54353 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1721	11.303175	192.168.29.120	49.44.235.212	TCP	66	59190 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

This command returns all the synchronization packets with value = 1