

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
```

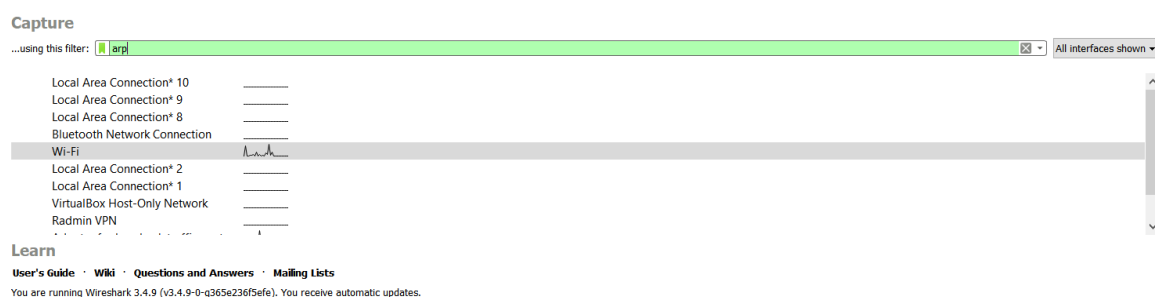
```
Wireless LAN adapter Wi-Fi:

    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.

# WIKIPEDIA

The Free Encyclopedia

English

6 383 000+ articles

日本語

1 292 000+ 記事

Español

1 717 000+ artículos

Deutsch

2 617 000+ Artikel

Русский

1 756 000+ статей

Français

2 362 000+ articles

中文

1 231 000+ 條目

Italiano

1 718 000+ voci

Português

1 074 000+ artigos

Polski

1 490 000+ hasel

ENQ

Read Wikipedia in your language

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.62? Tell 192.168.29.1
2	2.176175	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.115? Tell 192.168.29.1
3	4.224676	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.224? Tell 192.168.29.1
4	31.463516	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
5	31.463535	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
6	71.357521	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
7	71.357533	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
8	78.412753	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.204? Tell 192.168.29.1
9	79.59432	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.204? Tell 192.168.29.1
10	80.418797	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.204? Tell 192.168.29.1
11	88.534158	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
12	88.534185	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
13	89.550559	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
14	89.550578	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
15	90.618657	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.11? Tell 192.168.29.1
16	92.984116	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.142? Tell 192.168.29.1
17	97.168062	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.62? Tell 192.168.29.1
18	99.252831	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.115? Tell 192.168.29.1
19	100.276...	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.115? Tell 192.168.29.1
20	101.505...	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.224? Tell 192.168.29.1
21	117.930...	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
22	117.930...	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
23	147.996...	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
24	147.996...	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
25	175.371...	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.204? Tell 192.168.29.1
26	176.360...	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
27	176.360...	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
28	176.370...	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.204? Tell 192.168.29.1
29	185.679...	Sercomm_0e:aa:33	Broadcast	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
30	185.679...	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba
31	186.490...	Sercomm_0e:aa:33	IntelCor_a5:13:ba	ARP	42	Who has 192.168.29.120? Tell 192.168.29.1
32	186.490...	IntelCor_a5:13:ba	Sercomm_0e:aa:33	ARP	42	192.168.29.120 is at 50:e0:85:a5:13:ba

> Frame 6: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF{9A26F4A2-063C-4748-B045-13109AD00842E}, id 0  
> Ethernet II, Src: Sercomm\_0e:aa:33 (30:49:50:2e:aa:33), Dst: IntelCor\_a5:13:ba (50:e0:85:a5:13:ba)  
> Address Resolution Protocol (request)

0000 50 e0 85 a5 13 ba 30 49 50 2e aa 33 08 06 00 01 P...01 P...3...  
0010 00 00 06 04 00 01 50 e0 85 a5 13 ba c0 a8 1d 01 .....01...  
0020 00 00 00 00 00 00 c0 a8 1d 78 .....X

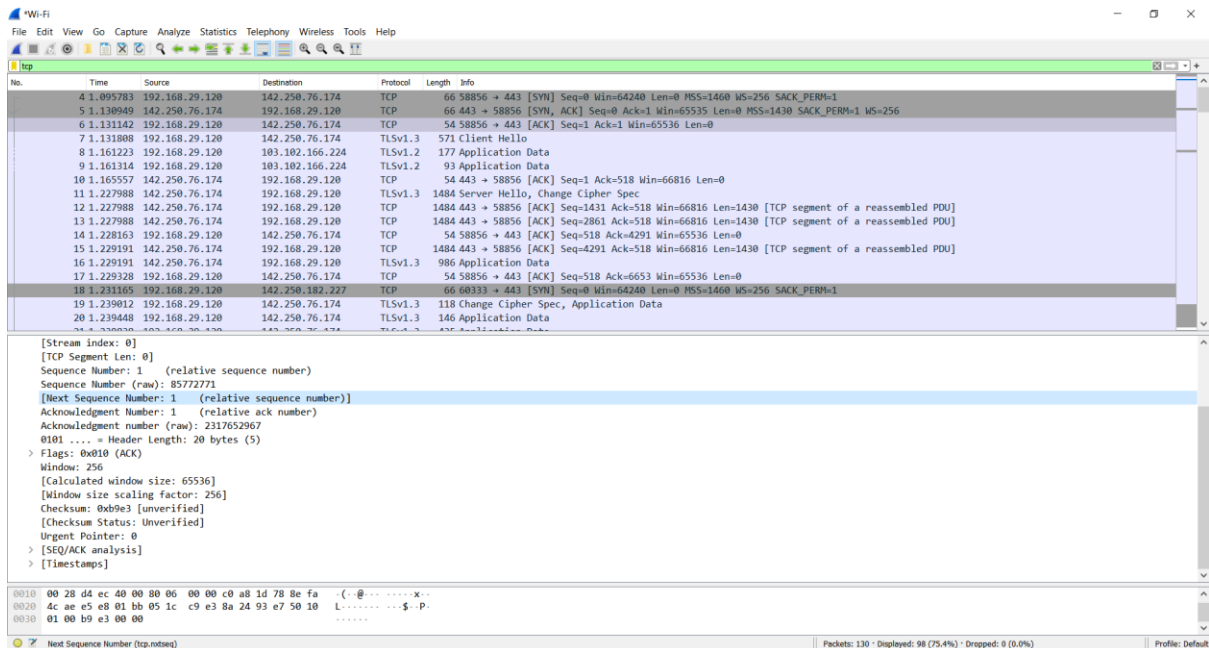
wireshark\_Wi-Fi-802.11.pcapng Packets: 32 · Displayed: 32 (100.0%) · Dropped: 0 (0.0%) Profile: Default

## 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.



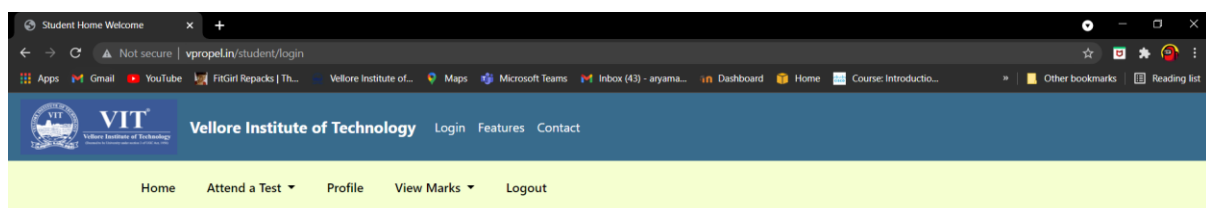
4	1.095783	192.168.29.120	142.250.76.174	TCP	66 58856 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	1.130949	142.250.76.174	192.168.29.120	TCP	66 443 → 58856 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
6	1.131142	192.168.29.120	142.250.76.174	TCP	54 58856 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0

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

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 Vpropel 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.



**Wish you Happy Coding (learning) with VPROPEL:)**  
[Click here for Demonstration Videos of Students login in VPROPEL](#)  
[Click here for PoD Discussion Videos](#)

**Problem of the Day**  
Practice and improve  
These questions are specially given to help you improve at programming **Each Problem is Available from 8pm of First Day to 8pm of Next Day**

[Solve today's question](#)

Author: M Janaki Meena Solved by: 175

**Problem of the Day Archives**  
Solution and editorials  
The solutions and editorials for the previously given daily challenges for the past one week.

[See Archive](#)

No.	Time	Source	Destination	Protocol	Length	Info
38	4.933530	192.168.29.120	52.66.12.101	HTTP	986	POST /student/login HTTP/1.1 (application/x-www-form-urlencoded)
57	5.100218	52.66.12.101	192.168.29.120	HTTP	59	HTTP/1.1 200 OK ([{'none':})

```

> Internet Protocol Version 4, Src: 192.168.29.120, Dst: 52.66.12.101
√ Transmission Control Protocol, Src Port: 62769, Dst Port: 80, Seq: 1, Ack: 1, Len: 932
  Source Port: 62769
  Destination Port: 80
  [Stream index: 14]
  [TCP Segment Len: 932]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 1096169100
  [Next Sequence Number: 933 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 912704778
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 513
  [Calculated window size: 513]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x2286 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
  TCP payload (932 bytes)
> Hypertext Transfer Protocol
√ HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "csrfmiddlewaretoken" = "rN2V9Byj3wrBRsQB4AudDVUXz5jTws4XcEYEFjyFrWTajK93wLtVEFn2cCGuz4Ub"
  > Form item: "studentRegNo" = "19bce1027"
  > Form item: "browser" = "1"
  √ Form item: "studentPassword" = "aryamanmishravpropel"
    Key: studentPassword
    Value: aryamanmishravpropel
  > Form item: "student_login" = ""

```

---

```

Form item: "csrfmiddlewaretoken" = "rN2V9Byj3wrBRsQB4AudDVUXz5jTws4XcEYEFjyFrWTajK93wLtVEFn2cCGuz4Ub"
Form item: "studentRegNo" = "19bce1027"
Form item: "browser" = "1"
Form item: "studentPassword" = "aryamanmishravpropel"
  Key: studentPassword
  Value: aryamanmishravpropel

```

75	7a	34	55	62	26	73	74	75	64	65	6e	74	52	65	67	uz4Ub&st	udentReg
4e	6f	3d	31	39	62	63	65	31	30	32	37	26	62	72	6f	No=19bce	1027&bro
77	73	65	72	3d	31	26	73	74	75	64	65	6e	74	50	61	wser=1&s	tudentPa
73	73	77	6f	72	64	3d	61	72	79	61	6d	61	6e	6d	69	ssword=a	ryamanmi
73	68	72	61	76	70	72	6f	70	65	6c	26	73	74	75	64	shravpro	pel&stud
65	6e	74	5f	6c	6f	67	69	6e	3d							ent_logi	n=

## 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.

```
C:\WINDOWS\system32>ping 1.1.1.1

Pinging 1.1.1.1 with 32 bytes of data:
Reply from 1.1.1.1: bytes=32 time=44ms TTL=50
Reply from 1.1.1.1: bytes=32 time=41ms TTL=50
Reply from 1.1.1.1: bytes=32 time=39ms TTL=50
Reply from 1.1.1.1: bytes=32 time=40ms TTL=50

Ping statistics for 1.1.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 39ms, Maximum = 44ms, Average = 41ms
```

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ICMP

No.	Time	Source	Destination	Protocol	Length	Info
3	2.806357	192.168.29.120	1.1.1.1	ICMP	74	Echo (ping) request id=0x0001, seq=33/8448, ttl=128 (reply in 4)
4	2.850661	1.1.1.1	192.168.29.120	ICMP	74	Echo (ping) reply id=0x0001, seq=33/8448, ttl=50 (request in 3)
6	3.821435	192.168.29.120	1.1.1.1	ICMP	74	Echo (ping) request id=0x0001, seq=34/8704, ttl=128 (reply in 7)
7	3.862831	1.1.1.1	192.168.29.120	ICMP	74	Echo (ping) reply id=0x0001, seq=34/8704, ttl=50 (request in 6)
28	4.839609	192.168.29.120	1.1.1.1	ICMP	74	Echo (ping) request id=0x0001, seq=35/8960, ttl=128 (reply in 30)
30	4.878782	1.1.1.1	192.168.29.120	ICMP	74	Echo (ping) reply id=0x0001, seq=35/8960, ttl=50 (request in 28)
35	5.855056	192.168.29.120	1.1.1.1	ICMP	74	Echo (ping) request id=0x0001, seq=36/9216, ttl=128 (reply in 36)
36	5.894914	1.1.1.1	192.168.29.120	ICMP	74	Echo (ping) reply id=0x0001, seq=36/9216, ttl=50 (request in 35)

Source

192.168.29.120

1.1.1.1

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 checkpageexec.microsoft.com	
86	66.227434	192.168.29.1	192.168.29.120	DNS	217	Standard query response 0x48ba A checkpageexec.microsoft.com CHAME wd-prod-ss.trafficmanager.net CHAME 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	66.457693	137.116.139.120	192.168.29.120	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message	
99	66.457802	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	66.606344	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	66.457693	137.116.139.120	192.168.29.120	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message	
99	66.457802	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	66.606344	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	67.109590	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

No.	Time	Source	Destination	Protocol	Length	Info
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
102	68.055989	142.251.42.67	192.168.29.120	TCP	93	[TCP Retransmission] 443 → 63449 [PSH, ACK] Seq=5636 Ack=1012 Win=67840 Len=39
299	68.055368	130.211.16.53	192.168.29.120	TLSv1.3	200	[TCP Previous segment not captured], Application Data
300	68.056114	192.168.29.120	130.211.16.53	TCP	66	[TCP Dup ACK 289#1] 53371 → 443 [ACK] Seq=1501 Ack=6142 Win=65280 Len=0 SLE=6753 SRE=6899
303	68.057335	130.211.16.53	192.168.29.120	TLSv1.3	120	[TCP Previous segment not captured], Application Data
304	68.058647	192.168.29.120	130.211.16.53	TCP	74	[TCP Dup ACK 289#2] 53371 → 443 [ACK] Seq=1501 Ack=6142 Win=65280 Len=0 SLE=7121 SRE=7187 SLE=6753 SRE=6899
308	68.062782	130.211.16.53	192.168.29.120	TLSv1.3	634	[TCP Fast Retransmission], Application Data, Application Data
311	68.064208	130.211.16.53	192.168.29.120	TCP	85	[TCP Out-Of-Order] 443 → 53371 [PSH, ACK] Seq=6722 Ack=1143 Win=67840 Len=31
312	68.064208	130.211.16.53	192.168.29.120	TCP	147	[TCP Out-Of-Order] 443 → 53371 [PSH, ACK] Seq=6899 Ack=1501 Win=68864 Len=93
313	68.064208	130.211.16.53	192.168.29.120	TCP	183	[TCP Out-Of-Order] 443 → 53371 [PSH, ACK] Seq=6992 Ack=1501 Win=68864 Len=129
909	69.772848	130.211.16.53	192.168.29.120	TLSv1.3	93	[TCP Previous segment not captured], Application Data
910	69.772997	192.168.29.120	130.211.16.53	TCP	66	[TCP Dup ACK 405#1] 53371 → 443 [ACK] Seq=1941 Ack=7298 Win=65536 Len=0 SLE=7663 SRE=7702
915	69.777767	130.211.16.53	192.168.29.120	TCP	331	[TCP Out-Of-Order] 443 → 53371 [PSH, ACK] Seq=7298 Ack=1941 Win=70912 Len=277
916	69.777767	130.211.16.53	192.168.29.120	TCP	142	[TCP Out-Of-Order] 443 → 53371 [PSH, ACK] Seq=7575 Ack=1941 Win=70912 Len=88
1341	70.666589	142.250.207.225	192.168.29.120	TCP	395	[TCP Spurious Retransmission] 443 → 49811 [PSH, ACK] Seq=7151 Ack=518 Win=66816 Len=341[Reassembly error, protocol TCP: New fragmen..
1342	70.666669	192.168.29.120	142.250.207.225	TCP	66	[TCP Dup ACK 1335#1] 49811 → 443 [ACK] Seq=582 Ack=7492 Win=65280 Len=0 SLE=7151 SRE=7492
1582	71.247814	3.224.52.149	192.168.29.120	TCP	66	[TCP Retransmission] 443 → 64407 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=256
2797	73.279969	3.224.52.149	192.168.29.120	TCP	66	[TCP Retransmission] 443 → 64407 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=256
4997	75.270984	49.44.226.108	192.168.29.120	TCP	643	[TCP Spurious Retransmission] 443 → 50633 [PSH, ACK] Seq=4381 Ack=518 Win=66816 Len=589[Reassembly error, protocol TCP: New fragmen..
4998	75.270984	49.44.226.108	192.168.29.120	TCP	643	[TCP Spurious Retransmission] 443 → 60300 [PSH, ACK] Seq=4381 Ack=518 Win=66816 Len=589[Reassembly error, protocol TCP: New fragmen..
5001	75.271139	192.168.29.120	49.44.226.108	TCP	66	[TCP Dup ACK 4932#1] 50633 → 443 [ACK] Seq=582 Ack=4970 Win=65536 Len=0 SLE=4381 SRE=4970
5002	75.271195	192.168.29.120	49.44.226.108	TCP	66	[TCP Dup ACK 4931#1] 60300 → 443 [ACK] Seq=582 Ack=4970 Win=65536 Len=0 SLE=4381 SRE=4970
8035	77.377783	3.224.52.149	192.168.29.120	TCP	66	[TCP Retransmission] 443 → 64407 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=256
15828	81.956501	172.217.167.234	192.168.29.120	TLSv1.3	93	[TCP Previous segment not captured], Application Data
15830	81.958052	172.217.167.234	192.168.29.120	TCP	400	[TCP Out-Of-Order] 443 → 49575 [PSH, ACK] Seq=5323 Ack=1537 Win=69632 Len=346
15831	81.958052	172.217.167.234	192.168.29.120	TCP	414	[TCP Out-Of-Order] 443 → 49575 [PSH, ACK] Seq=5669 Ack=1537 Win=69632 Len=360
21898	85.607586	3.224.52.149	192.168.29.120	TCP	66	[TCP Retransmission] 443 → 64407 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=256
25528	94.835570	192.168.29.120	214.250.193.68	TCP	54	[TCP Retransmission] 56440 → 443 [FIN, ACK] Seq=582 Ack=4209 Win=65536 Len=0
25529	94.835571	192.168.29.120	216.58.203.3	TCP	54	[TCP Retransmission] 64057 → 443 [FIN, ACK] Seq=582 Ack=4326 Win=65536 Len=0
25531	94.840958	192.168.29.120	161.69.226.70	TCP	54	[TCP Retransmission] 60246 → 443 [FIN, ACK] Seq=593 Ack=185 Win=65280 Len=0
27206	101.787.3.224.52.149	192.168.29.120	TCP	66	[TCP Retransmission] 443 → 64407 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 SACK_PERM=1 WS=256	

## e. !(arp or icmp or dns)

No.	Time	Source	Destination	Protocol	Length	Info
40	42.597024	192.168.29.120	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
41	43.605481	192.168.29.120	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
42	44.032549	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
43	44.619264	192.168.29.120	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
45	45.877881	192.168.29.224	239.255.255.250	SSDP	330	NOTIFY * HTTP/1.1
46	45.878179	192.168.29.224	239.255.255.250	SSDP	339	NOTIFY * HTTP/1.1
47	45.914402	192.168.29.224	239.255.255.250	SSDP	386	NOTIFY * HTTP/1.1
48	45.956734	192.168.29.224	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
49	45.996549	192.168.29.224	239.255.255.250	SSDP	396	NOTIFY * HTTP/1.1
50	45.999838	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
51	46.037355	192.168.29.224	239.255.255.250	SSDP	384	NOTIFY * HTTP/1.1
52	46.078237	192.168.29.224	239.255.255.250	SSDP	330	NOTIFY * HTTP/1.1
53	46.118231	192.168.29.224	239.255.255.250	SSDP	339	NOTIFY * HTTP/1.1
54	46.158270	192.168.29.224	239.255.255.250	SSDP	386	NOTIFY * HTTP/1.1
55	46.199796	192.168.29.224	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
56	46.241091	192.168.29.224	239.255.255.250	SSDP	396	NOTIFY * HTTP/1.1
57	46.281798	192.168.29.224	239.255.255.250	SSDP	384	NOTIFY * HTTP/1.1
58	46.321337	192.168.29.224	239.255.255.250	SSDP	330	NOTIFY * HTTP/1.1
59	46.363546	192.168.29.224	239.255.255.250	SSDP	339	NOTIFY * HTTP/1.1
60	46.403528	192.168.29.224	239.255.255.250	SSDP	386	NOTIFY * HTTP/1.1
61	46.444880	192.168.29.224	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
62	46.484627	192.168.29.224	239.255.255.250	SSDP	396	NOTIFY * HTTP/1.1
63	46.526727	192.168.29.224	239.255.255.250	SSDP	384	NOTIFY * HTTP/1.1
64	46.567536	192.168.29.224	239.255.255.250	SSDP	330	NOTIFY * HTTP/1.1
65	46.607485	192.168.29.224	239.255.255.250	SSDP	339	NOTIFY * HTTP/1.1
66	46.647800	192.168.29.224	239.255.255.250	SSDP	386	NOTIFY * HTTP/1.1
67	46.688369	192.168.29.224	239.255.255.250	SSDP	394	NOTIFY * HTTP/1.1
68	46.729182	192.168.29.224	239.255.255.250	SSDP	396	NOTIFY * HTTP/1.1
69	46.769969	192.168.29.224	239.255.255.250	SSDP	384	NOTIFY * HTTP/1.1
70	48.128296	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
71	50.381650	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
73	52.429027	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
74	54.477466	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
76	56.525635	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
77	58.778468	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
78	60.826516	192.168.29.224	192.168.29.255	UDP	84	53842 → 5775 Len=42
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

## 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.

The screenshot shows the Wireshark interface with a TCP stream selected. The top pane displays the raw packet data in hexadecimal and ASCII. The middle pane shows the packet details, including the TLS handshake process. The bottom pane shows the packet list with a filter applied to the selected stream.

No.	Time	Source	Destination	Protocol	Length	Info
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	54	51493 → 443 [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	54	51493 → 443 [ACK] Seq=1661 Ack=212 Win=525312 Len=1460 [TCP segment of a reassembled PDU]
93	66.391901	137.116.139.120	192.168.29.120	TCP	54	51493 → 443 [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	54	51493 → 443 [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

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	157.248.198.15	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	10.855300	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	10.870100	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	59198 → 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