NAME : Deepesh Vikas Patil **MIS NO :** 112215055 Lab Assignment-1 (Group-2)

Q1. (a) To specify the **number of echo requests** to send with ping, we use the **'-c**' flag. We mention the number of requests after using the above flag. For eg. **$** ping www.hackthebox.com -c 10
The ping is requested 10 times, when the above command is run.

(b). To **set the time interval between each ping**, we use the flag '**-i'** and then mention the time interval (in seconds or in fraction). For Eg. **$** ping www.hackthebox.com -c 5 -i 0.2

(c). To ping without waiting for a reply, we can simply type '**$** ping www.hackthebox.com'.
For a non-superuser in Linux, there is limit to how many maximum number of ping requests you can send. However, if we are using CMD/Powershell on windows, it only permits 4 ping requests to a non-superuser.

(d). In order to **set the size of the payload/size in bytes** we can use the **'-s'** flag followed by the packet size, as shown in the following commands: **$** ping www.hackthebox.com -s 128
Suppose if we set the packet size to 'n' bytes, the total size of the packet will be 'n+ICMP header size + IP header size' Bytes. Usually, the ICMP header size = 8 and the IP header size is '20' for IPv4 and '40' for IPv6. For eg., if we consider the packet size to be 32 Bytes, the total packet size will be 60 Bytes (IPv4).

Q2. List of Hosts:
www.google.com | www.hackthebox.com | www.github.com | www.youtube.com | www.amazon.in | www.amazon.com

(a).

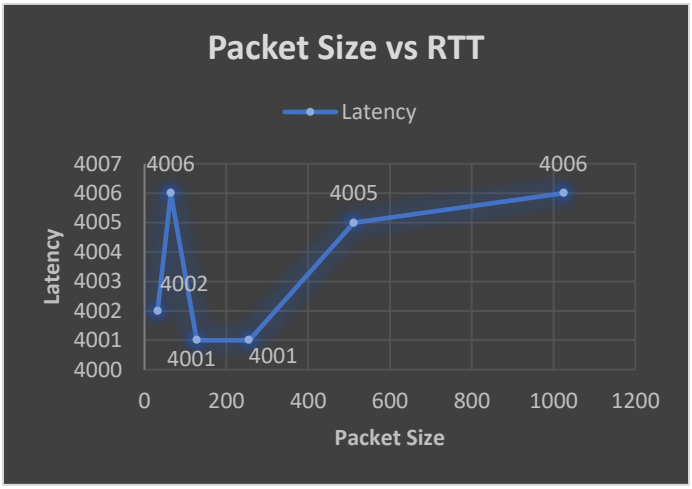
HOST	Avg RTT (11:50 am)	Avg RTT (2:00 am)	Avg RTT (9:00 pm)
www.google.com	9.173 ms	86.131 ms	34.813 ms
www.github.com	13.993 ms	31.391 ms	22.626 ms
www.hackthebox.com	10.234 ms	109.010 ms	43.583 ms
www.youtube.com	10.910 ms	64.527 ms	33.103 ms
www.amazon.in	151.853 ms	269.378 ms	172.788 ms
www.amazon.com	222.398 ms	242.558 ms	300.307 ms

The RTT is affected by the geographical distance between the sender and host. As the distance increases, the RTT also increases because of increased hop counts. As we can see in the above table, the 'www.amazon.in' takes less time to respond, and compared to 'www.amazon.com' (in most cases).

(b). For all the above hosts, there was **0%** (no) packet loss. This could be due to an excellent Wi-Fi network or no Network congestion. Packet Loss occurs because of **network congestion**, **hardware issues**, **software bugs**, and a number of other factors that cause dropped packets during data transmission.

(c). Host : www.amazon.in

PACKET SIZE	RTT(ms)
32 B	4002
64 B	4006
128 B	4001
256 B	4001
512 B	4005
1024 B	4006



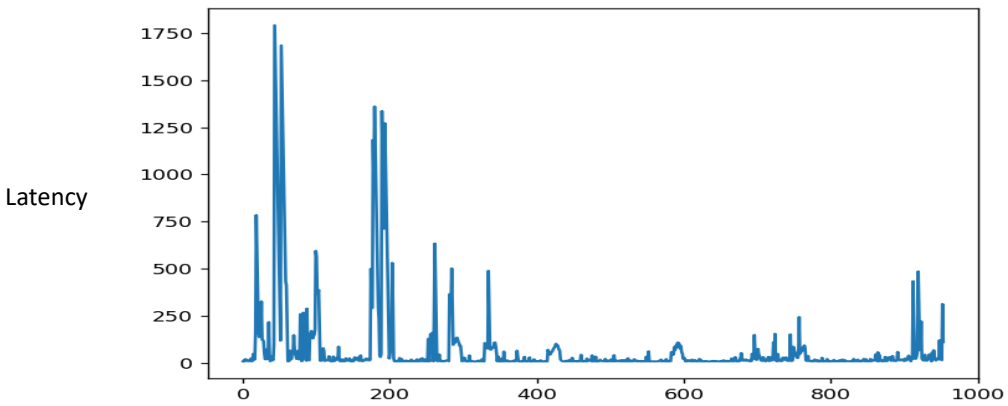
(d). The **RTT** depends on factors like **size of the packets**, **time of the day, network congestion**, etc. As the size of the packets increase, the transmission rate gradually decreases. This causes packets to take longer transmission time, especially over long distances. Network Congestion is another factor that affects the RTT of a packet. When more users access a particular website, network congestion increases the RTT of data packets decrease. Thus at night time the RTT of data packets from a website is less as compared to the same website in the day time. This implies, the time of the day also affects the RTT of data packets.

(3). Host: www.github.com
I have used Python for the analysis of the text files (ping1.txt and ping2.txt).

(a). Packet Loss for Command-1: 4.6%
Packet Loss for Command-2: 6.8%

(b). Command - 1: Minimum: 7.615 ms, Average: 72.986 ms, Maximum: 1792.153 ms, Median: 13.5 ms
Command – 2: Minimum: 7.645 ms, Average: 84.896 ms, Maximum: 1259.674 ms, Median: 13.4 ms

(c).



(d). Cmd1 sends packets with normal patterns, whereas the 'ff00' is a hexadecimal pattern. Due to the hexadecimal pattern, the Average Ping Latency increases as compared to the normal pattern packets.
Thus, cmd1 is faster as compared to cmd2.

Q4. (a) The '**ifconfig**' helps to **configure displays the network interface(s)** information of Unix/Linux OS. There is a similar command used in windows, which works the same, '**ipconfig'**. There are two interfaces in most of the cases, the 'eth0' and 'lo'. In both interfaces, we get 'Flag' which shows its status, weather it is active/running/up or down/inactive. Next, it shows the 'MTU' or 'maximum Transmission Unit' that shows the max size of the packet that can be transferred via that network. Then it shows the IPv4 (IP Addr., Netmask Addr., Broadcast Addr.) and IPv6 (Link-Local Addr.) configuration of the device. It also shows the hardware MAC Addr. of the interface along with transmission length. Lastly, it shows the 'Receive' & 'Transmit' Statistics.

(b). The 'ifconfig' command can be used for various purposes such as displaying and configuring the network interface information or to enable/disable it. Here are some of the functions of the 'ifconfig'.

- **$** 'ifconfig' – To display the Network Interface Information
- **$** 'sudo' ifconfig eth0 <IP ADDR.> netmask 255.255.255.0' – To configure the network interfaces
- **$** 'sudo ifconfig eth0 <UP/DOWN>' – To enable/disable the network interface
- **$** 'sudo ifconfig eth0 hw ether 00:11:22:33:44:55' – For MAC address spoofing

(c). Here are some of the following options that can be used along with 'ifconfig'.

- '-a' :- it displays all interfaces that are currently available, even if they are down
- '-s' :- it displays a shortlist of statistical analysis of the network interfaces.
- '-v' :- it can be more verbose for some error conditions
- '<interface>' :- it can be used to assign more addresses

Q5. (a). '**netstat'** is a utility tool used to print network connections, routing tables, interface statistics, masquerade connections, and multicast membership.

(b). To see all the established tcp connections, we include the following options with the 'netstat' command: '-t', '-a'
Eg.: **$** sudo netstat -t -a grep 'ESTABLISHED'

```
deep@DESKTOP-BO5BPJS:/mnt/c/Users/Deepesh Patil$ sudo netstat -t -a  grep 'ESTABLISHED'
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:domain       0.0.0.0:*               LISTEN
```

(c). 'netstat -r' shows the Kernal IP routing table. This command shows the following information:
- Destination IP address - IP address of the next hop-up gateway - the subnet mask of host
- routing flag, i.e. (UG by default, but U is the route is up) - the network interface associated with the route
- TCP window size (used to control the amount of unacknowledged data sent) - Initial RTT
- Maximum segment size, i.e. the maximum amount of data that can be sent in a single TCP segment (except TCP header).

(d). The option '-i' can be used with netstat to view the kernel interface table, which lists all the network interface.
Eg.: **$** netstat -i
There are two interfaces. On my desktop, there are two interfaces i.e. 'eth0' and 'lo'.

(e). To see the statistics of all UDP connections, we use the netstat command with option '-su'.

$ netstat -su

```
deep@DESKTOP-BO5BPJS:/mnt/c/Users/Deepesh Patil$ netstat -su
IcmpMsg:
    OutType3: 7
Udp:
    12 packets received
    7 packets to unknown port received
    0 packet receive errors
    16 packets sent
    0 receive buffer errors
    0 send buffer errors
    IgnoredMulti: 60
UdpLite:
IpExt:
    InBcastPkts: 60
    InOctets: 609275
    OutOctets: 59676
    InBcastOctets: 5295
    InNoECTPkts: 610
```

(f). The **Loop-back interface** (represented as 'lo' in netstat) is a special network interface which have several functions such as ensuring the proper functioning of the network, local testing and diagnostics.

- The loop-back interface allows Linux system to address itself (localhost addressing).
- It can also be used in testing applications locally by simulating network comms without the need of an actual network.
- The loop-back interface can also be used for network services testing and diagnostics.
- The loop-back interface is also used for IPC (Inter-Process Communications)
- It can also be used for DNS testing.

Q6. (a). The traceroute tool is used to trace and print the route taken by the data packets sent from an IP network to a host.

(b).

HOSTS	NO. OF HOPS (1:45 pm)	NO. OF HOPS (4:00 pm)	NO. OF HOPS (1:10 am):
www.google.com	9	10	10
www.github.com	>30	>30	>30
www.hackthebox.com	11	11	11
www.youtube.com	9	10	9
www.amazon.in	>30	>30	>30
www.amazon.com	>30	>30	>30

First few hops are common for almost every host.

(c). The route of the same host changes depending on the time of the day. The data packets are sent to the host via a particular route and these routes are based on **OSPF** (Open Shortest path First) or **BGP** (Border Gateway Protocol). These protocols designate a route according to various network conditions such as network load/traffic and topology change. Since at different time the load on the network varies, the route as well varies.

(d). There are various reasons for why sometimes traceroute does not find complete paths. Some routers reject/blocks the ICMP (Internet Control Message Protocol) packets (used for traceroute) due to their **firewall settings**. Also, high level of **network traffic** may lead to prioritization of other packets than the ICMP packets, which leads to delayed or no response. Due to **limitations in the TTL** field of the ICMP packets, the **distance** may also lead to failure of route tracing. If a **network is under maintenance** or if a **router is misconfigured**, the routers fail to respond to the ICMP packets. Some **security policies** may be implemented by the network admin to restrict the complete view of the route.

(e). Since 'ping' and 'traceroute' work on different protocols ('ping' works with **ICMP**, whereas 'traceroute' can also work with **UDP**), the 'traceroute' is able to find a route to the host, even if the host fails to respond to the ping. The 'ping' generates/sends the ICMP echo requests, whereas the 'traceroute' works by sending the data packets to the host and sometimes since the respond exceeds the TTL of the ping, it fails to generate response, but the data packets can reach the host.

Q7. (a). We can see the full ARP table with the following command: **$** sudo arp
The table shows six columns, the first one is the **IP address** of the device. The second column shows the **hardware type** (HWtype: ether, arcnet, hdlc, etc.). The third column shows the **MAC** (Media Access Control) **address** of the device, used for WoL, security and access control. The next column shows the '**flag**' associated with entry ('C' for complete, 'W' for stale, 'U' for unreachable, etc.). The 'mask' column shows the **network mask** and the 'Iface' column shows the **network interface** associated with the ARP table (eth0, wlan0, etc.)

(b). To add an entry in the ARP table, we use the command: **$** sudo arp -s <IP ADDR.> <MAC ADDR.> -i <INTERFACE>
To delete an entry, we use: **$** sudo arp -ad <IP ADDR.> -i <INTERFACE>

```
Address                  HWtype  HWaddress           Flags Mask            Iface
127.0.0.2                ether   00:11:22:33:44:55   CM                    eth0
127.0.0.1                ether   00:11:22:33:44:55   CM                    eth0
```

(c). ARP operates at the **Layer 2** (data link layer) of the OSI model which supports local network communication. When two devices having same subnet communicate, they **do not require routing**, and the ARP resolves IP to MAC address within the same subnet. When the communication involves devices from different subnets, it often requires routing through a router. By this, they are able to forward packets between different subnets. Thus, we cannot add an entry of an IP with different subnet.