Cyber Security

SEZG681/SSZG681

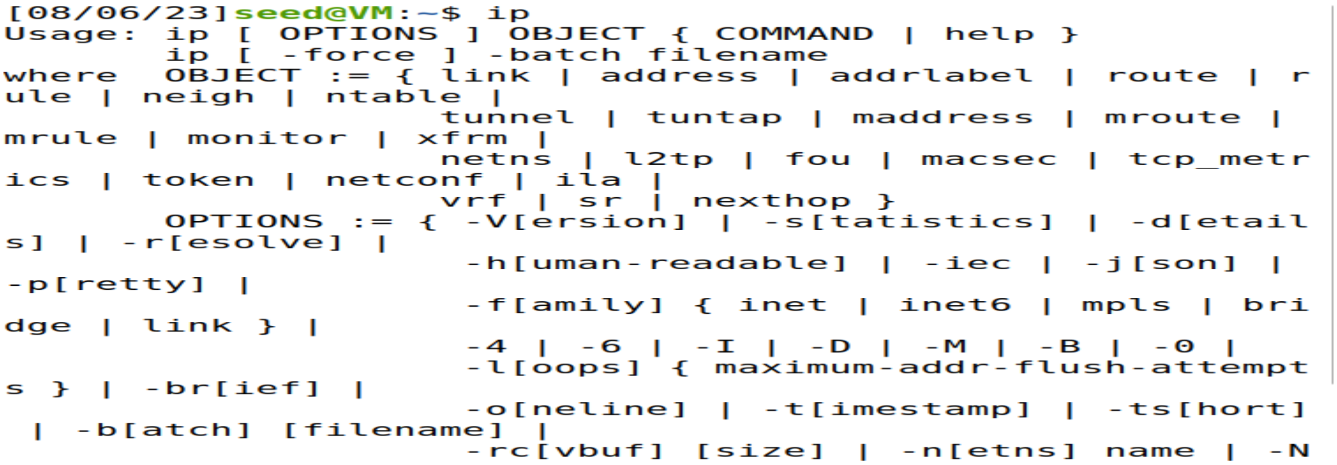
Lab 0 – Networking Commands

In this labsheet, we will go through a few networking commands that will be helpful to perform the attacks that we are going to study in the upcoming labs. There are a few commands that should always be in your sysadmin toolbox. You can take internet help to have deeper knowledge of these commands. Let us study the commands one by one

1. ip: The `ip` command is one of the basic commands that every network administrator should know. It is used in setting up new systems and assigning IPs to troubleshooting existing systems. The `ip` command can show address information, manipulate routing, plus display network various devices, interfaces, and tunnels.

The syntax is as follows:

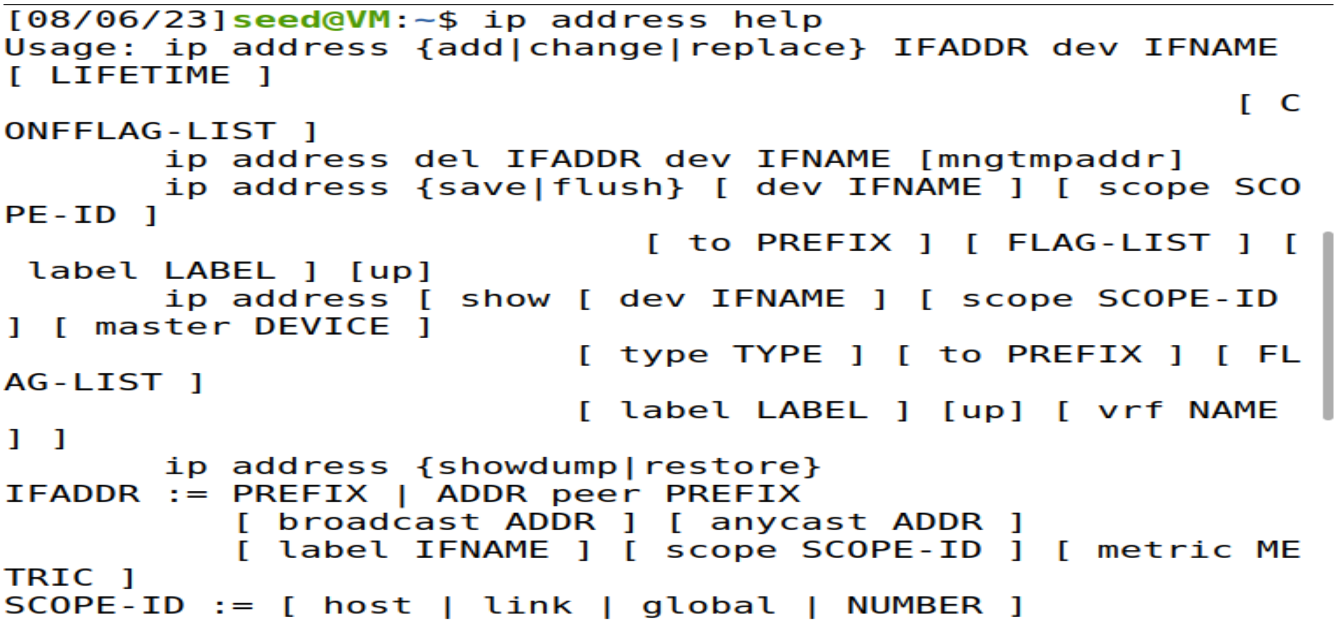
ip <OPTIONS> <OBJECT> <COMMAND>



```
[08/06/23]seed@VM:~$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | r
ule | neigh | ntable |
                   tunnel | tuntap | maddress | mroute |
mrule | monitor | xfrm |
                   netns | l2tp | fou | macsec | tcp_metr
ics | token | netconf | ila |
                   vrf | sr | nexthop }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etail
s] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] |
-p[retty] |
                    -f[amily] { inet | inet6 | mpls | bri
dge | link } |
                    -4 | -6 | -I | -D | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempt
s } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort]
 | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -N
```

“help” is an important utility that we can use to get more information about a particular object. To get help with respect to a particular object, write the following command: ip address help

Here are some common use cases for the ip command. Execute the following commands in your terminal and see what happens



```
[08/06/23]seed@VM:~$ ip address help
Usage: ip address {add|change|replace} IFADDR dev IFNAME
[ LIFETIME ]
                                                      [ C
ONFFLAG-LIST ]
       ip address del IFADDR dev IFNAME [mngtmpaddr]
       ip address {save|flush} [ dev IFNAME ] [ scope SCO
PE-ID ]
                                 [ to PREFIX ] [ FLAG-LIST ] [
 label LABEL ] [up]
       ip address [ show [ dev IFNAME ] [ scope SCOPE-ID
] [ master DEVICE ]
                          [ type TYPE ] [ to PREFIX ] [ FL
AG-LIST ]
                          [ label LABEL ] [up] [ vrf NAME
] ]
       ip address {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
          [ broadcast ADDR ] [ anycast ADDR ]
          [ label IFNAME ] [ scope SCOPE-ID ] [ metric ME
TRIC ]
SCOPE-ID := [ host | link | global | NUMBER ]
```

a. To show the IP addresses assigned to an interface on your machine:

```
ip address show
```

b. To assign an IP to an interface, for example, enps03 (this device name could be different):

```
ip address add 192.168.1.254/24 dev enps03
```

c. To delete an IP on an interface:

```
ip address del 192.168.1.254/24 dev enps03
```

d. Alter the status of the interface by bringing the interface eth0 online:

```
ip link set eth0 up
```

e. Alter the status of the interface by bringing the interface eth0 offline:

```
ip link set eth0 down
```

and many more

2. ifconfig: The ifconfig command was/is a staple in many sysadmin's tool belt for configuring and troubleshooting networks. It has since been replaced by the ip command discussed above.

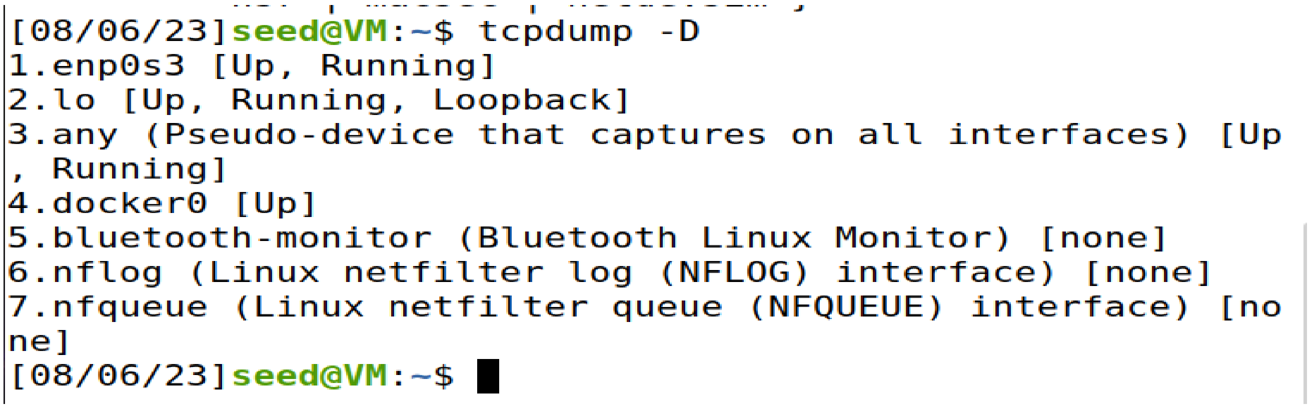
3. tcpdump: The tcpdump command is designed for capturing and displaying packets.

You can install tcpdump with the command below:

```
dnf install -y tcpdump
```

Before starting any capture, you need to know which interfaces `tcpdump` can use. You will need to use sudo or have root access in this case. Use the following command to know the interfaces

```
tcpdump -D
```



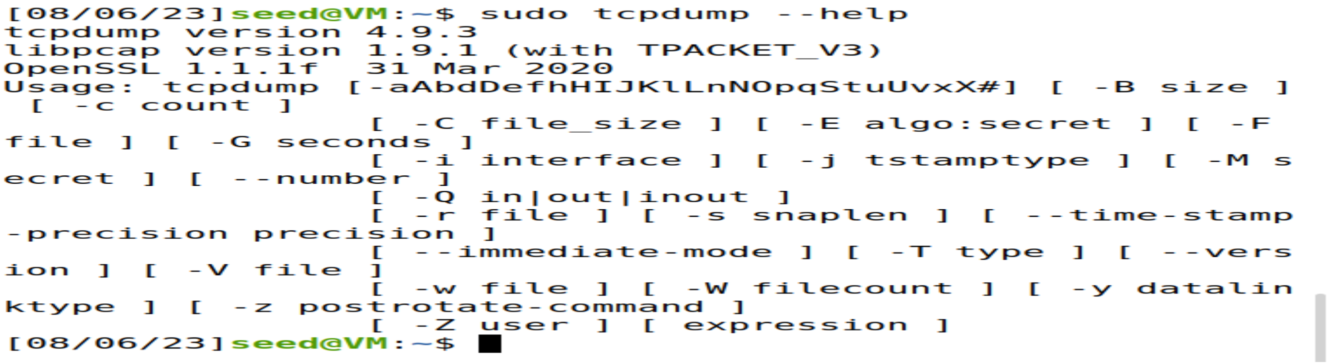
```
[08/06/23]seed@VM:~$ tcpdump -D
1.enp0s3 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up
, Running]
4.docker0 [Up]
5.bluetooth-monitor (Bluetooth Linux Monitor) [none]
6.nflog (Linux netfilter log (NFLOG) interface) [none]
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [no
ne]
[08/06/23]seed@VM:~$ █
```

You can use the following commands to **capture traffic on vmnet1 (you can change it)**, you can initiate that with `tcpdump -i vmnet1` sample output:

```
tcpdump -i vmnet1

tcpdump -i vmnet1 -c 10
```

You can take **help command to know the specific parameters** used with this command. You can play with these parameters and take internet help to get more specific information.



```
[08/06/23]seed@VM:~$ sudo tcpdump --help
tcpdump version 4.9.3
libpcap version 1.9.1 (with TPACKET_V3)
OpenSSL 1.1.1f  31 Mar 2020
Usage: tcpdump [-aAbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ]
 [ -c count ]
                [ -C file_size ] [ -E algo:secret ] [ -F
file ] [ -G seconds ]
                [ -i interface ] [ -j tstamptype ] [ -M s
ecret ] [ --number ]
                [ -Q in|out|inout ]
                [ -r file ] [ -s snaplen ] [ --time-stamp
-precision precision ]
                [ --immediate-mode ] [ -T type ] [ --vers
ion ] [ -V file ]
                [ -w file ] [ -W filecount ] [ -y datalin
ktype ] [ -z postrotate-command ]
                [ -Z user ] [ expression ]
[08/06/23]seed@VM:~$ █
```

a. Capture traffic to and from one host

You can filter out traffic coming from a specific host. For example, to find traffic coming from and going to 8.8.8.8 (this ip is just an example), use the command:

```
tcpdump -i vmnet1 -c 10 host 8.8.8.8
```

For traffic coming from 8.8.8.8, use:

```
tcpdump -i vmnet1 src host 8.8.8.8
```

For outbound traffic going to 8.8.8.8, use:

```
tcpdump -i vmnet1 dst host 8.8.8.8
```

b. Capture traffic to and from a network

You can also capture traffic to and from a specific network using the command below:

```
tcpdump -i vmnet1 net 10.1.0.0 mask 255.255.255.0
```

or

```
tcpdump -i vmnet1 net 10.1.0.0/24
```

You can also filter based on either source or destination.

Based on the source (traffic coming from):

```
tcpdump -i vmnet1 src net 10.1.0.0/24
```

Based on the destination (traffic going to):

```
tcpdump -i vmnet1 dst net 10.1.0.0/24
```

c. Capture traffic to and from port numbers:

Capture only DNS port 53 traffic:

```
tcpdump -i vmnet1 port 53
```

For a specific host:

```
tcpdump -i vmnet1 host 8.8.8.8 and port 53
```

To capture only HTTPS traffic:

```
tcpdump -i vmnet1 -c 10 host www.google.com and port 443
```

To capture all port except port 80 and 25:

```
tcpdump -i vmnet1 port not 53 and not 25
```

4. netstat: The `netstat` tool for printing network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. This utility is part of the net-tool package, as is `ifconfig`. In the new iproute2 package, the `ss` tool is used to achieve the same objectives.

If `netstat` is not found on your system, install it with this command:

```
dnf install net-tools
```

The primary usage of `netstat` is without any parameters:

```
netstat
```

For advanced usage, expand the `netstat` command with options:

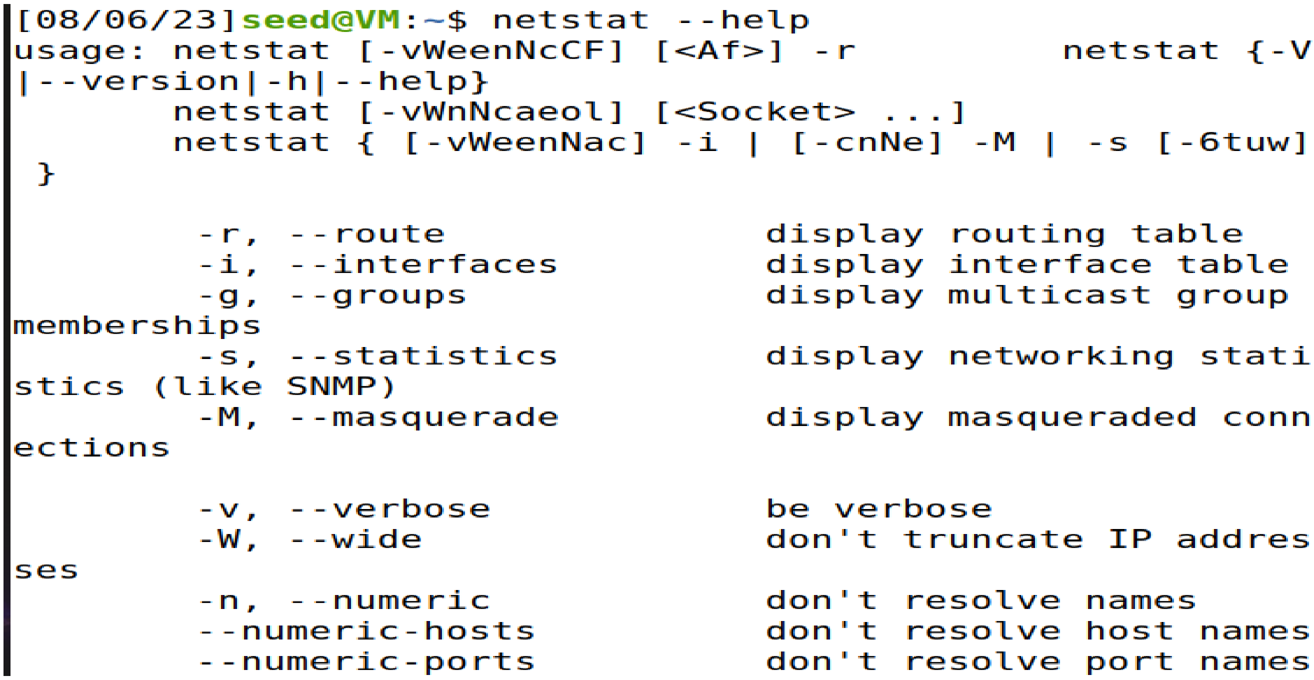
```
netstat <options>
```

Or list the options one by one:

```
netstat <option 1> <option 2> <option 3>
```

You can use the following command to know the available options:

netstat –help



```
[08/06/23]seed@VM:~$ netstat --help
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V
|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw]
 }

        -r, --route                display routing table
        -i, --interfaces           display interface table
        -g, --groups               display multicast group
memberships
        -s, --statistics           display networking stati
stics (like SNMP)
        -M, --masquerade           display masqueraded conn
ections

        -v, --verbose              be verbose
        -W, --wide                 don't truncate IP addres
ses
        -n, --numeric              don't resolve names
        --numeric-hosts            don't resolve host names
        --numeric-ports            don't resolve port names
```

a. To list all ports and connections regardless of their state or protocol, use:

```
netstat -a
```

b. List all TCP ports by running:

```
netstat -at
```

c. List all UDP ports with:

```
netstat -au
```

d. To return a list of only listening ports for all protocols, use:

```
netstat -l
```

e. List all listening TCP ports with:

```
netstat -lt
```

f. Return only listening UDP ports by running:

```
netstat -lu
```

g. To list UNIX listening ports, use:

```
netstat -lx
```

h. Display statistics for all ports regardless of the protocol with:

```
netstat -s
```

i. List statistics for TCP ports only with:

```
netstat -st
```

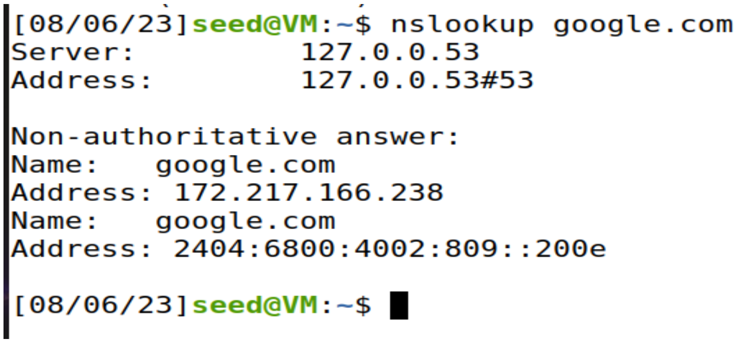
j. To find a process that is using a particular port number, run:

```
netstat -an | grep ‘:<port number>’
```

5. nslookup: Use the `nslookup` utility to query Internet name servers interactively. Use it to perform DNS queries and receive domain names or IP addresses, or any other specific DNS records.
Consider the following common examples.

To find the A record of a domain:

```
nslookup google.com
```



```
[08/06/23]seed@VM:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.166.238
Name:   google.com
Address: 2404:6800:4002:809::200e

[08/06/23]seed@VM:~$ 
```

To check the NS records of a domain:

```
nslookup -type=ns example.com
```

To find all of the available DNS records of a domain:

```
nslookup -type=any example.com
```

To check the use of a specific DNS server (in this case, query using the specific nameserver ns1.nsexample.com):

```
nslookup example.com ns1.nsexample.com
```

Checking DNS A records to see the IPs of a domain is a common practice, but sometimes you need to verify if an IP address is related to a specific domain. For that purpose, you need a reverse DNS lookup.

```
nslookup 10.20.30.40
```

6. ping: Ping is a tool that verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages is displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution.

Simple `ping` commands take only one parameter: The hostname or the host IP address that you want to verify. A simple `ping` example is just like below:

```
ping google.com
```

Note: Most of well known sites doesn’t allow to ping their server for various reasons: **Security and Privacy, Traffic Reduction, Firewall and Network Configuration, Load Balancing and Redundancy, DDoS Protection, and may be some Legacy Reasons.** As a result, you won’t get reply from their servers.

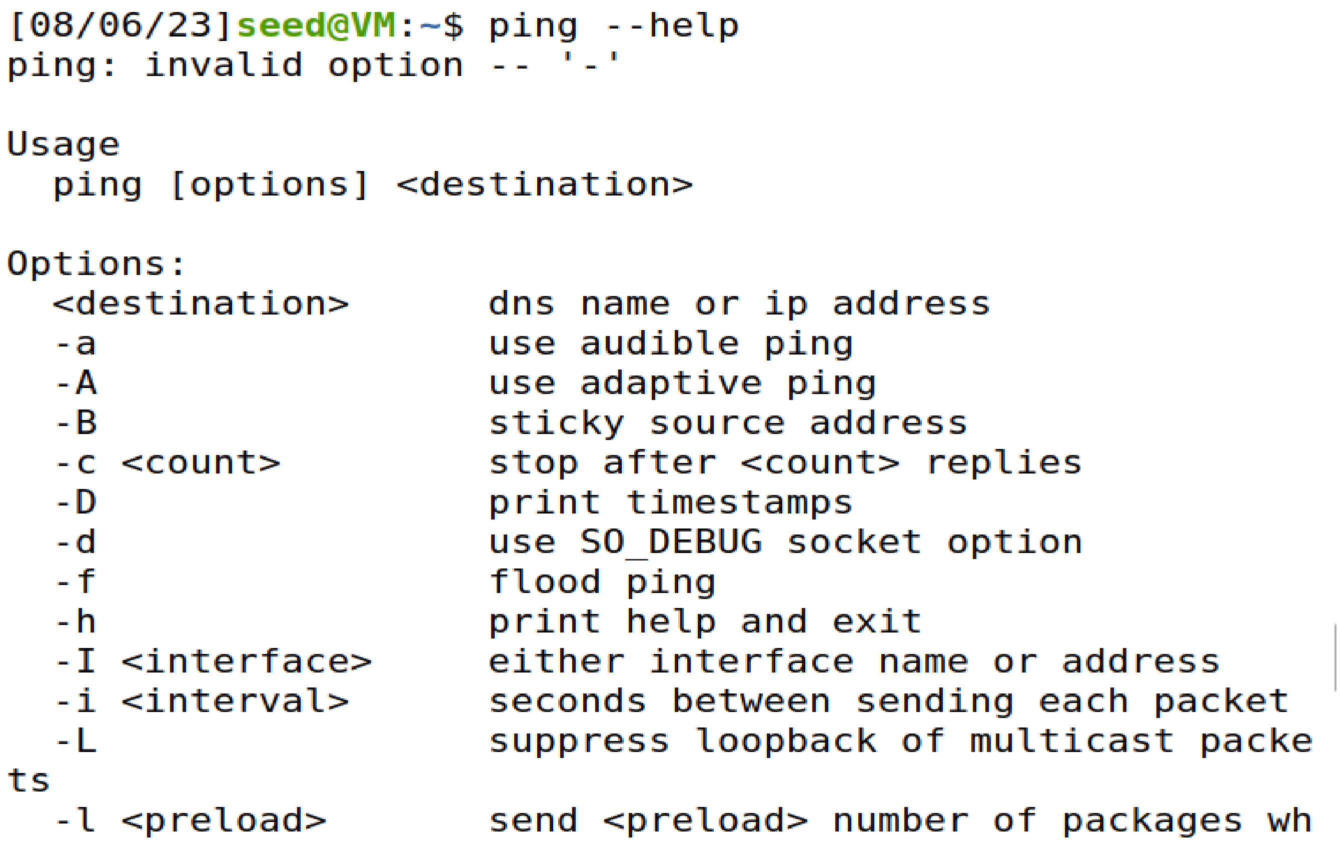
You need to stop the `ping` command by pressing **CTRL+C**. Otherwise, it will `ping` until you stop it. After every `ping` command, it will display a summary report with the following information:

- Min: Minimum time that it takes to get a response from the host that has been pinged from your end.
- Avg: Average time that it takes to get a response from the host that has been pinged from your end.
- Max: Maximum time that it takes to get a response from the host that has been pinged from your end.

Also, you will see TTL, which stands for Time To Live. Ping uses a numerical TTL value to attempt to reach a given host computer via the route path. This is also known as the hop limit.

You can use the following command to know various options used with this command:

ping -help



```
[08/06/23]seed@VM:~$ ping --help
ping: invalid option -- '-'

Usage
  ping [options] <destination>

Options:
  <destination>      dns name or ip address
  -a                 use audible ping
  -A                 use adaptive ping
  -B                 sticky source address
  -c <count>         stop after <count> replies
  -D                 print timestamps
  -d                 use SO_DEBUG socket option
  -f                 flood ping
  -h                 print help and exit
  -I <interface>     either interface name or address
  -i <interval>      seconds between sending each packet
  -L                 suppress loopback of multicast packe
ts
  -l <preload>       send <preload> number of packages wh
```

Normally, when you run a simple `ping` command without passing any additional parameters, Linux will `ping` that host for an infinite amount of time. If you want to `ping` a host ten times, use the following command:

```
ping -c 10 google.com
```