**1.Write a Python program to implement the Caesar cipher using Substitution technique.**

```
# Expl: Implement Python program to illustrate Caesar Cipher Technique

def encrypt(text, s):
    result = ""
    # Traverse text
    for char in text:
        # Encrypt uppercase characters
        if char.isupper():
            result += chr((ord(char) + s - 65) % 26 + 65)
        # Encrypt lowercase characters
        elif char.islower():  # Added elif to explicitly check for 
lowercase
            result += chr((ord(char) + s - 97) % 26 + 97)
        else:
            # If it's not an alphabet, keep it as is (spaces, numbers, 
symbols)
            result += char
    return result


# Check the function
text = input("Enter text to encrypt: ")
try:
    s = int(input("Enter shift (0-25): "))
    print("Cipher: " + encrypt(text, s))
except ValueError:
    print("Enter a valid integer between 0 to 25")
```

```
Enter text to encrypt: Hello World
Enter shift (0-25): 4
Cipher: Lipps Asvph
```

**2.Write a Python program to implement and analysis of RSA.**

**python -m pip install sympy**

```
# simple_rsa_sympy.py
from sympy import isprime, gcd, mod_inverse

#key generation
def generate_keys(p: int, q: int, e: int):
    if not (isprime(p) and isprime(q)):
        raise ValueError("Both numbers must be prime.")
    if p == q:
        raise ValueError("p and q should not be the same.")

    n = p * q
    phi_n = (p - 1) * (q - 1)

    if gcd(e, phi_n) != 1:
        raise ValueError(f"e = {e} is not coprime with phi(n) = {phi_n}.
Choose a different e.")

    d = mod_inverse(e, phi_n)

    print("\nComputed values:")
    print(f"n = {n}")
    print(f"phi(n) = {phi_n}")
    print(f"Private key d = {d}")

    return (e, d, n)

# encryption function
def encrypt(message, e, n):
    if not (0 <= message < n):
        raise ValueError("Message integer must satisfy 0 <= message < n.")
    return pow(message, e, n)

#decryption function
def decrypt(cipher, d, n):
    return pow(cipher, d, n)



#main
if __name__ == "__main__":
    try:
        # User input for primes and exponent
        p = int(input("Enter a prime number p: "))
        q = int(input("Enter a different prime number q: "))
        e = int(input("Enter public exponent e (coprime with phi(n)): "))

        # Key generation
        e, d, n = generate_keys(p, q, e)

        #message input
        message = int(input("\nEnter integer message to encrypt (0 <= m <
n): "))

        #encrypt
        cipher = encrypt(message, e, n)
        print(f"Encrypted message (integer): {cipher}")

        #decrypt
        decrypted = decrypt(cipher, d, n)
        print(f"Decrypted integer: {decrypted}")

    except ValueError as ve:
        print(f"Error: {ve}")
```

```
Enter a prime number p: 5
Enter a different prime number q: 7
Enter public exponent e (coprime with phi(n)): 11

Computed values:
n = 35
phi(n) = 24
Private key d = 11

Enter integer message to encrypt (0 <= m < n): 2
Encrypted message (integer): 18
Decrypted integer: 2
```

**3.Write a Python program to implement the Diffie-Hellman Key Exchange algorithm.**

```
# Diffie-Hellman Key Exchange Algorithm

# Step 1: Get public values from user
n = int(input("Enter a prime number (n): "))
g = int(input("Enter a primitive root modulo n (g): "))

# Step 2: Get private keys from Alice and Bob
x = int(input("Alice, enter your private key (x): "))
y = int(input("Bob, enter your private key (y): "))

# Step 3: Calculate public keys
A = pow(g, x, n)  # (g^x) % n
B = pow(g, y, n)  # (g^y) % n

print(f"\nAlice's Public Key (A) = {A}")
print(f"Bob's Public Key (B) = {B}")

# Step 4: Exchange public keys and compute shared secret
k1 = pow(B, x, n)  # (B^x) % n
k2 = pow(A, y, n)  # (A^y) % n

print(f"\nAlice's computed shared secret = {k1}")
print(f"Bob's computed shared secret = {k2}")

# Step 5: Verify
if k1 == k2:
    print(f"\nKey exchange successful! Shared secret = {k1}")
else:
    print("\nKey exchange failed!")
```

```
Enter a prime number (n): 23
Enter a primitive root modulo n (g): 5
Alice, enter your private key (x): 6
Bob, enter your private key (y): 15

Alice's Public Key (A) = 8
Bob's Public Key (B) = 19

Alice's computed shared secret = 2
Bob's computed shared secret = 2

Key exchange successful! Shared secret = 2
```

**4.Write a Python program to implement MD5 Algorithm**

```
# md5_example.py
import hashlib

def generate_md5(message: str) -> str:
    """Generate MD5 hex digest for the provided message string."""
    # Create MD5 hash object
    md5_hash = hashlib.md5()
    # Convert string to bytes and update hash object
    md5_hash.update(message.encode('utf-8'))
    # Return hexadecimal digest
    return md5_hash.hexdigest()

if __name__ == "__main__":
    # Take input from user
    text = input("Enter text to hash using MD5: ").strip()
    md5_result = generate_md5(text)
    print("Original Text:", text)
    print("MD5 Hash:", md5_result)
```

```
Enter text to hash using MD5: hello world
Original Text: hello world
MD5 Hash: 5eb63bbbe01eeed093cb22bb8f5acdc3
```

**5.Write a Python program to implement SHA Algorithm**

```
# Program: Implement SHA-1 Algorithm using Python

import hashlib

# Function to generate SHA-1 hash
def generate_sha1(message):
    # Create SHA-1 hash object
    sha1_hash = hashlib.sha1()
    # Convert string to bytes and update hash object
    sha1_hash.update(message.encode('utf-8'))
    # Return hexadecimal digest
    return sha1_hash.hexdigest()

# Take input from user
text = input("Enter text to hash using SHA-1: ")
sha1_result = generate_sha1(text)

print("Original Text:", text)
print("SHA-1 Hash:", sha1_result)
```

```
Enter text to hash using SHA-1: hello world
Original Text: hello world
SHA-1 Hash: 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed
```

**6.Explain and show the execution of basic network commands with different options for each commands:**
**ifconfig, ping, netstat, Whois, dig, nslookup, traceroute, host,arp, hostname**

Software: Ubuntu 20. 0 4 LTS

Sudo apt install net-tools

1. ifconfig :
Linux ifconfig stands for interface configurator. It is one of the most basic commands used in network inspection. ifconfig is used to initialize an interface, configure it with an IP address, and enable or disable it. It is also used to display the route and the network interface.
Basic information displayed upon using ifconfig are:
1. IP address
2. MAC address
3. MTU(Maximum Transmission Unit)
To get all the details using ifconfig
Ifconfig
Using this command, you can get details of a specific interface. This is shown below.
Commands:
**ifconfig eno1**
**ifconfig lo**

ip command:
The ip addr command without any subcommands shows the network interface information, including the associated IP addresses:
Syntax:
ip addr

**ip -4 addr** (IP Command is used to show network interface information)
**ip -6 addr** To set the size of MTU
By default, MTU has a size of 1500. This can be however set externally by the user using
ifconfig.
Syntax: Ifconfig eno1 mtu xxxx

2. Netstat:
netstat(Network Statistics) is the command that is used to display routing table,
connection information, the status of ports. i.e. it gives details information about
network activity on Linux System.
a. **netstat -r** Using -r flag will display the routing table

b. **netstat –a** use -a flag to show listening and non-listening sockets(TCP+ UDP)
c. **netstat –at** to show all TCP socket
d. **netstat –l** to show all active ports
e. **netstat –u** to show all UDP socket

3 .Whois
The whois command displays information about a website's record. You may get all the
information about a website regarding its registration and owner's information.
Syntax:
whois <websiteName>
Example:
**whois javatpoint.com**
4.dig
dig command stands for Domain Information Groper. It is used for retrieving information about
DNS name servers. It is basically used by network administrators. It is used for verifying and
troubleshooting DNS problems and to perform DNS lookups.
Syntax:

dig [server] [name] [type]

**dig geeksforgeeks.org**

5.ping :
Ping (Packet Internet Groper) command is the best way to test connectivity between two nodes. Whether it is Local Area Network (LAN) or Wide Area Network (WAN).
the ping command is used to ensure that a computer can communicate to a specified
device over the network. The pings command sends Internet Control Message Protocol(ICMP) Echo Request messages in the form of packets to the destination
computer and waits in order to get the response back. Once the packets are received
by the destined computer, it starts sending the packets back. This command keeps
executing until it si interrupted.
Ping uses ICMP (Internet Control Message Protocol) to communicate to other devices.
ping command provides details such as:  number of packets transmitted
number of packets received
time taken by the packet to return

ping command in generally used for the following purposes:  measuring the time taken by the packets to return to determine the speed of the connection
to make sure that the network connection between the host and the destined computer can be established
Example:  **ping google.com**
**ping 216.58.203.142**
6. traceroute :
This utility uses the ICMP protocol and finds the hops involved in reading the destination
server. It also shows the time it takes between hops. This command is used to get the route
of a packet. i.e. to determine the path along which a packet travels. It also returns the
number of hops taken by the packet to reach the destination. This command prints to the
console, a list of hosts through which the packet travels in order to the destination.
Example:
**traceroute www.google.com**
**traceroute 4.2.2.2**

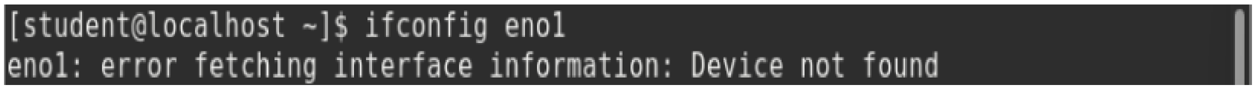
-4 Option: Use ip version 4 i.e. use IPv4
Syntax:
traceroute -4 google.com
-4 Option: Use ip version 6 i.e. use IPv6
Syntax:
traceroute -6 google.com

7. nslookup:
nslookup command queries the DNS in order to fetch the IP address or the domain
name from DNS records.
Example:
**nslookup www.facebook.com**
8. route:
route command also shows and manipulates the ip routing table
To see the default routing table in Linux, type the following command.
# **route**

9. host:
host command is used to find a domain name associated with the IP address or find an
IP address associated with the domain name. The returned IP address is either IPv4 or
IPv6.
Example:

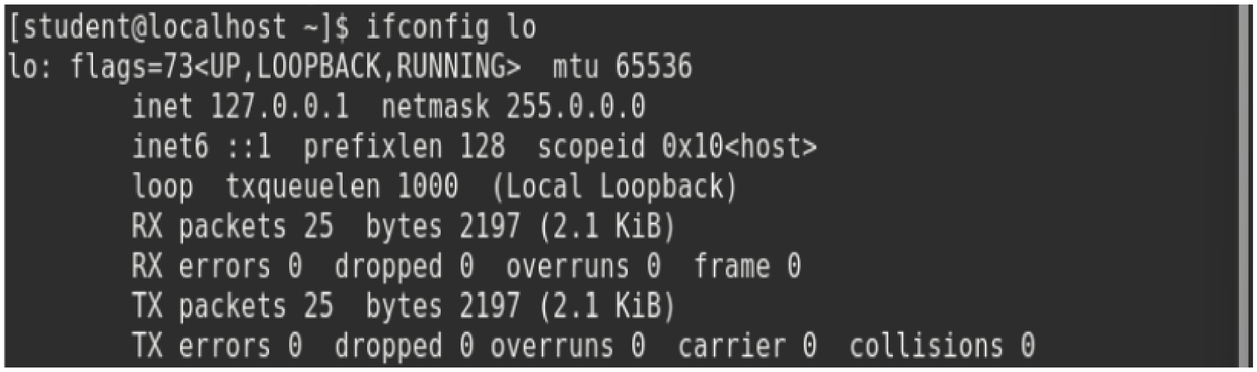
```
host www.google.com
host 31.13.78.35

10. hostname To check and set the hostname of the server.
Example:
hostname
11. arp:
ARP(Address Resolution Protocol) command is used to display and modify
ARP cache, which contains the mapping of IP address to MAC address.

The system’s TCP/IP stack uses ARP in order to determine the MAC
address associated with an IP address.
ARP (Address Resolution Protocol) is useful to view/add the contents of
the
kernel’s ARP tables. To see the default table use the command as.
# arp -e
```

**ifconfig eno1** - ifconfig (interface configurator) - configure network interfaces in Unix-like systems. It allows you to display, configure, and manage network interface settings like IP addresses, subnet masks, and interface status.



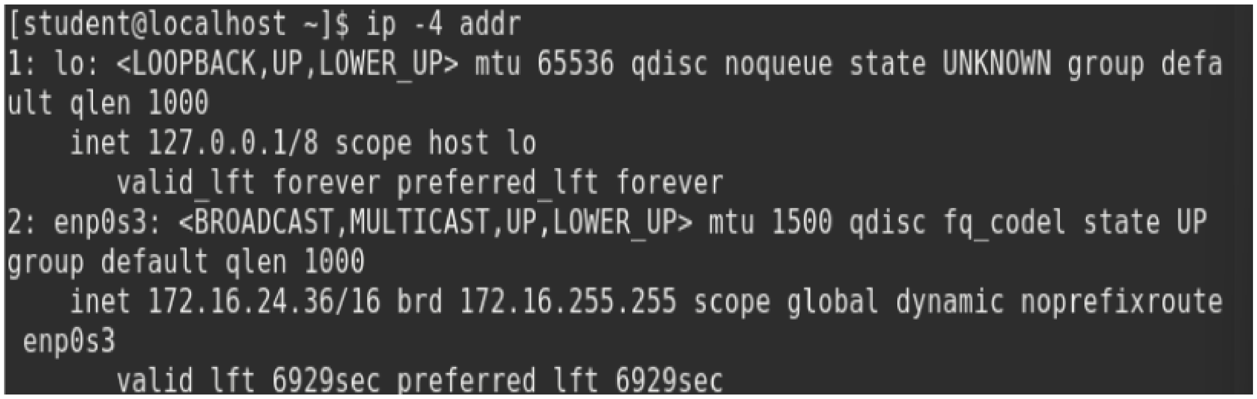
```
[student@localhost ~]$ ifconfig eno1
eno1: error fetching interface information: Device not found
```

**ifconfig lo**

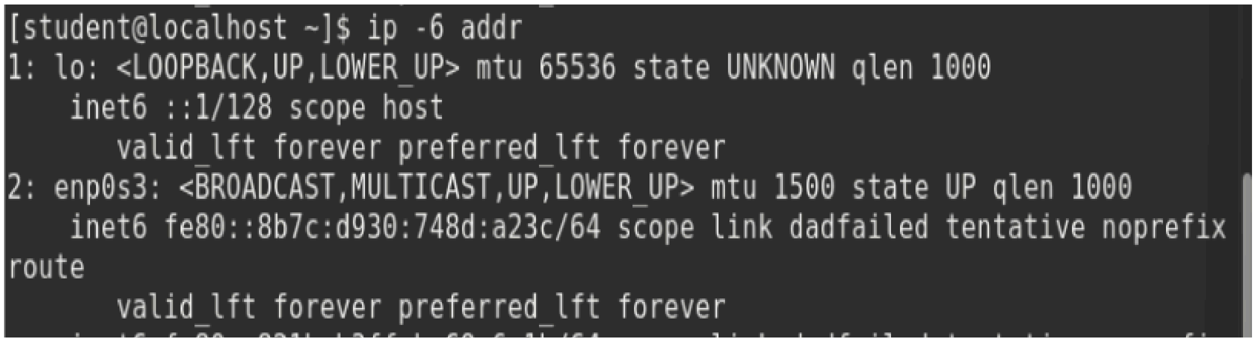


```
[student@localhost ~]$ ifconfig lo
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 25  bytes 2197 (2.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 2197 (2.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

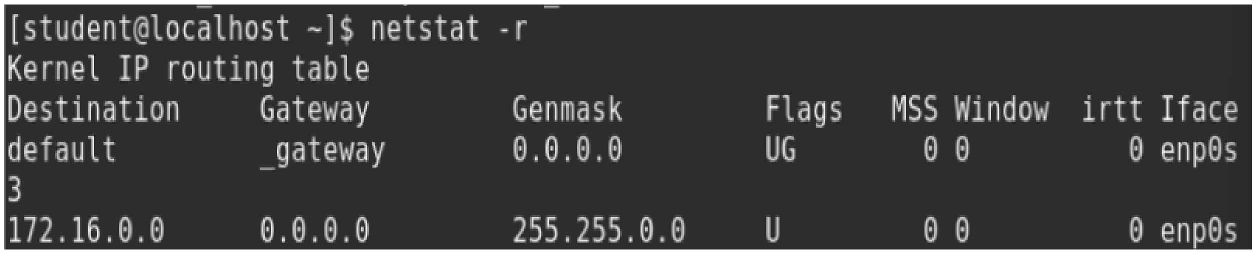
**ip -4 addr** - manage and retrieve information about network connections, specifically related to the Internet Protocol (IP)


```
[student@localhost ~]$ ip -4 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defa
ult qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    inet 172.16.24.36/16 brd 172.16.255.255 scope global dynamic noprefixroute
 enp0s3
       valid_lft 6929sec preferred_lft 6929sec
```

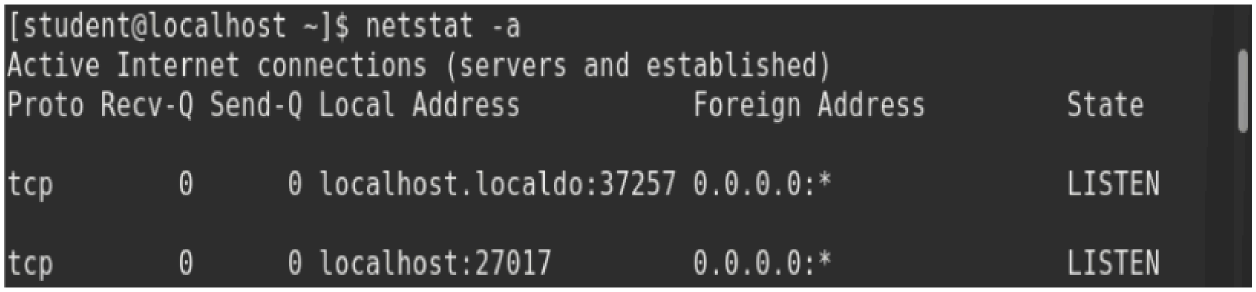
**ip -6 addr**


```
[student@localhost ~]$ ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 fe80::8b7c:d930:748d:a23c/64 scope link dadfailed tentative noprefix
route
       valid_lft forever preferred_lft forever
```

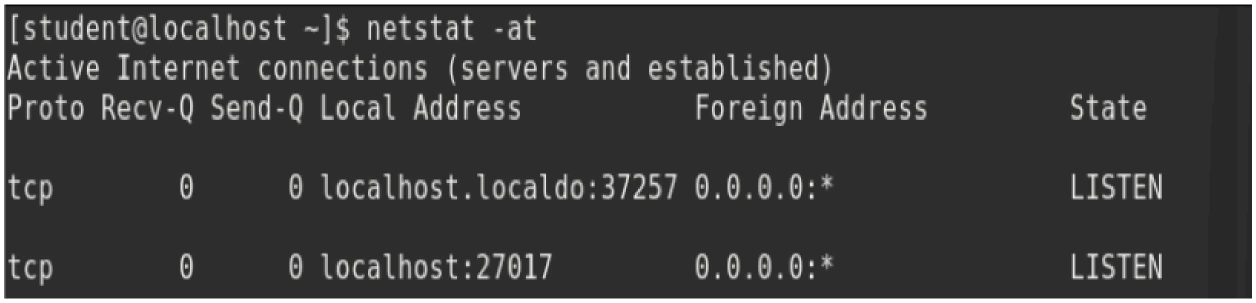
**netstat -r** - Using -r flag will display the routing table


```
[student@localhost ~]$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp0s
3
172.16.0.0      0.0.0.0         255.255.0.0     U         0 0          0 enp0s
```

**netstat –a** - use -a flag to show listening and non-listening sockets(TCP+ UDP)

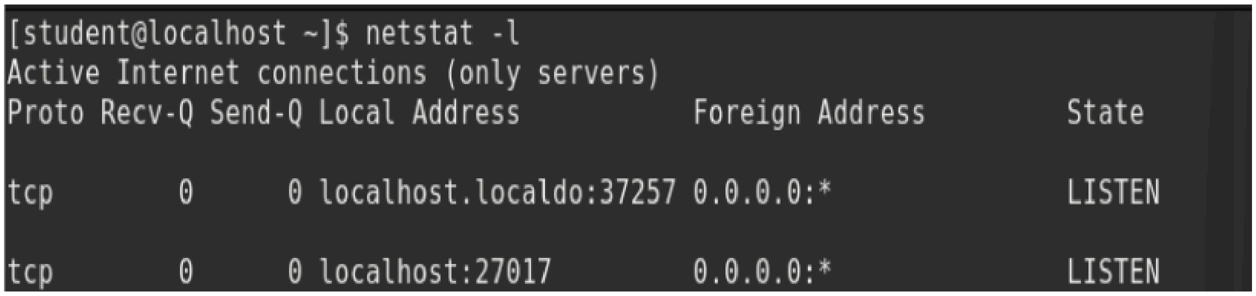

```
[student@localhost ~]$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 localhost.localdo:37257 0.0.0.0:*               LISTEN

tcp        0      0 localhost:27017         0.0.0.0:*               LISTEN
```

**netstat –at** - to show all TCP socket



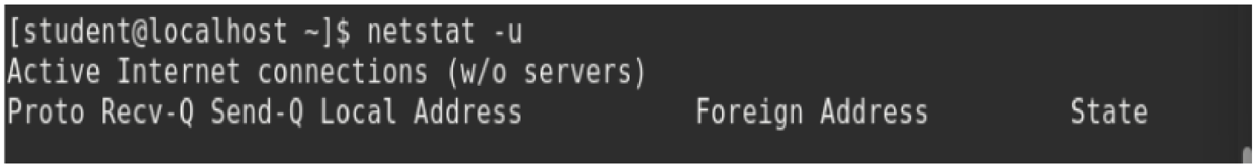
```
[student@localhost ~]$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 localhost.localdo:37257 0.0.0.0:*               LISTEN

tcp        0      0 localhost:27017         0.0.0.0:*               LISTEN
```

**netstat –l** - to show all active ports



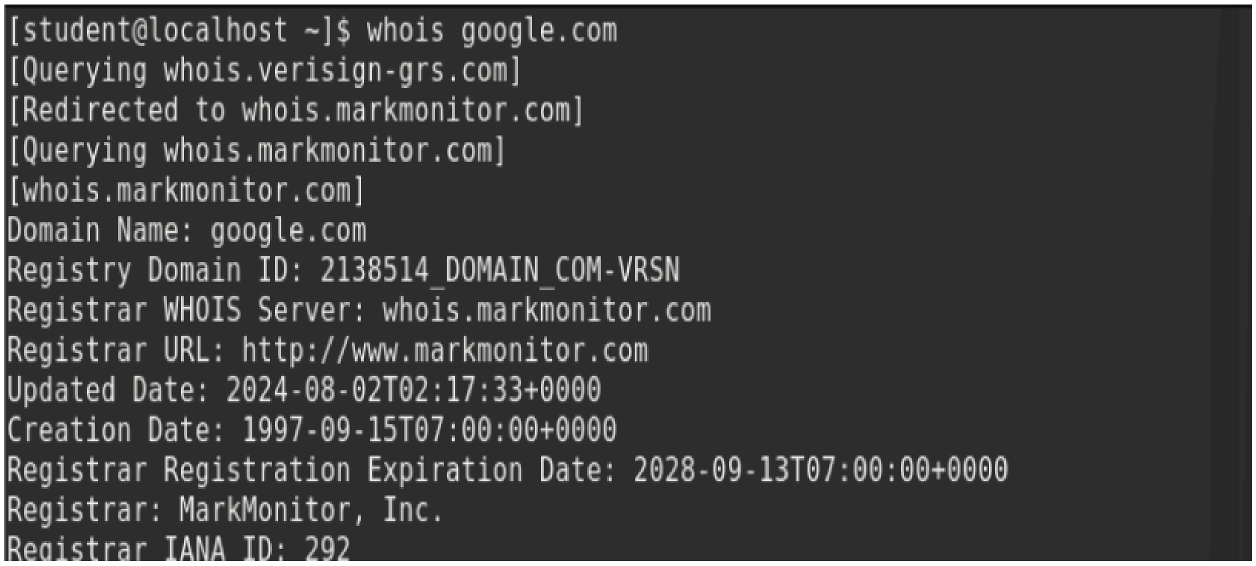
```
[student@localhost ~]$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 localhost.localdo:37257 0.0.0.0:*               LISTEN

tcp        0      0 localhost:27017         0.0.0.0:*               LISTEN
```

**netstat –u** - to show all UDP socket



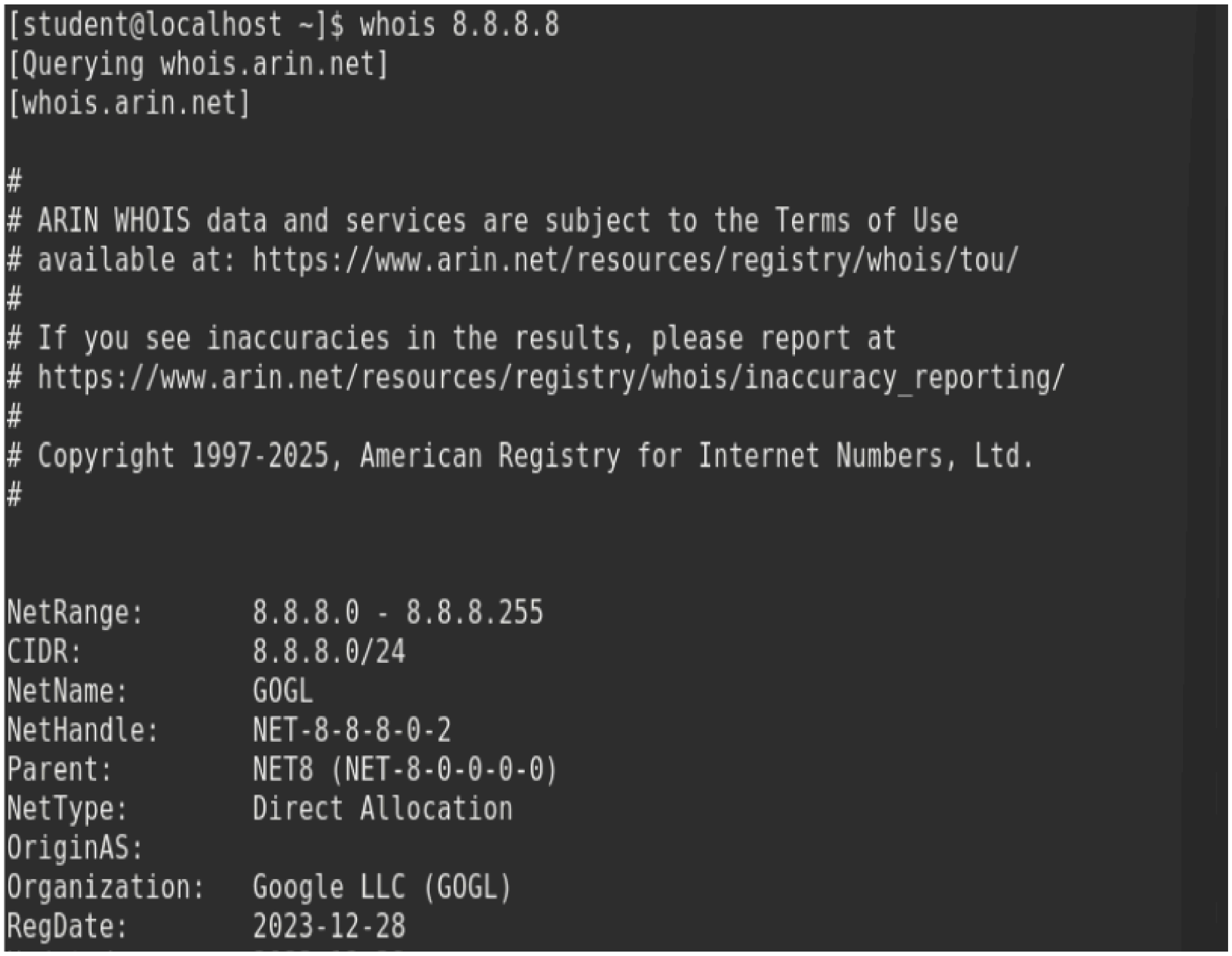
```
[student@localhost ~]$ netstat -u
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

**whois** [www.google.com](www.google.com) - displays information about a website's record i.e., all the information about a website regarding its registration and owner's information.



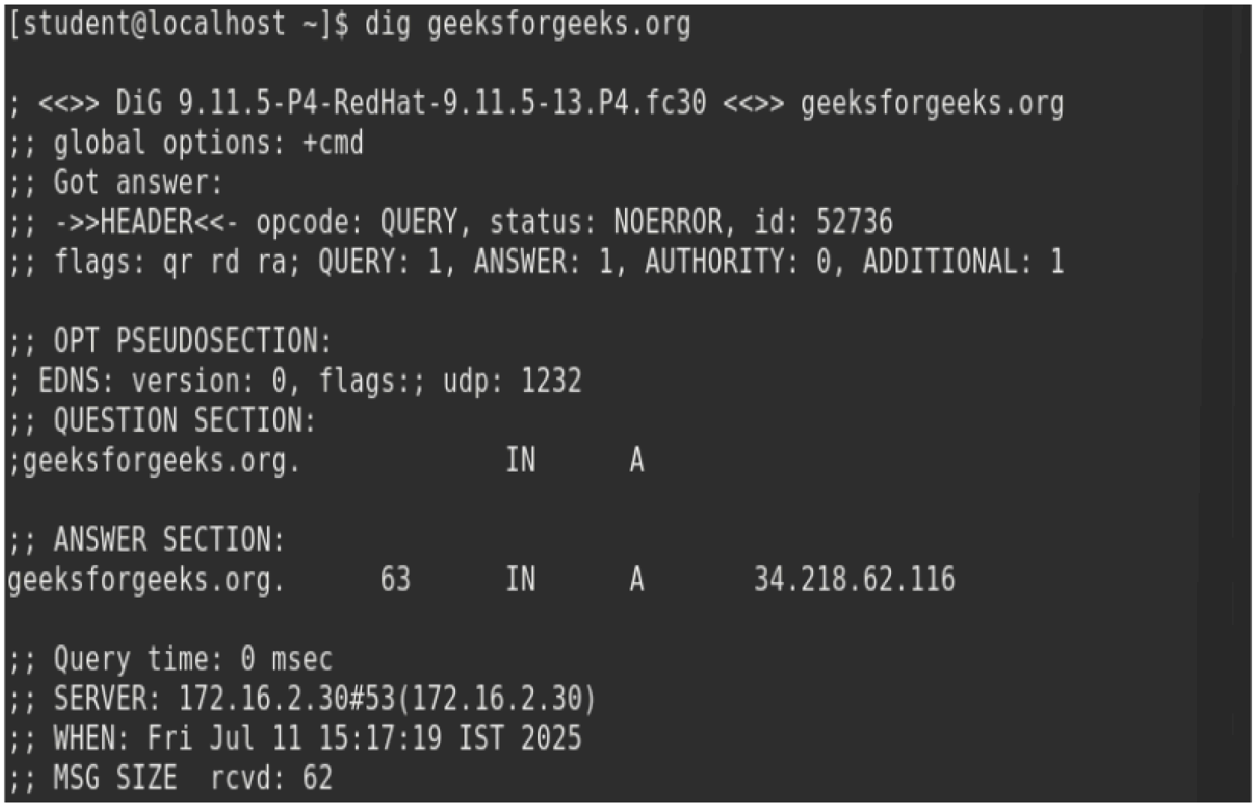
```
[student@localhost ~]$ whois google.com
[Querying whois.verisign-grs.com]
[Redirected to whois.markmonitor.com]
[Querying whois.markmonitor.com]
[whois.markmonitor.com]
Domain Name: google.com
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2024-08-02T02:17:33+0000
Creation Date: 1997-09-15T07:00:00+0000
Registrar Registration Expiration Date: 2028-09-13T07:00:00+0000
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
```

whois 8.8.8.8



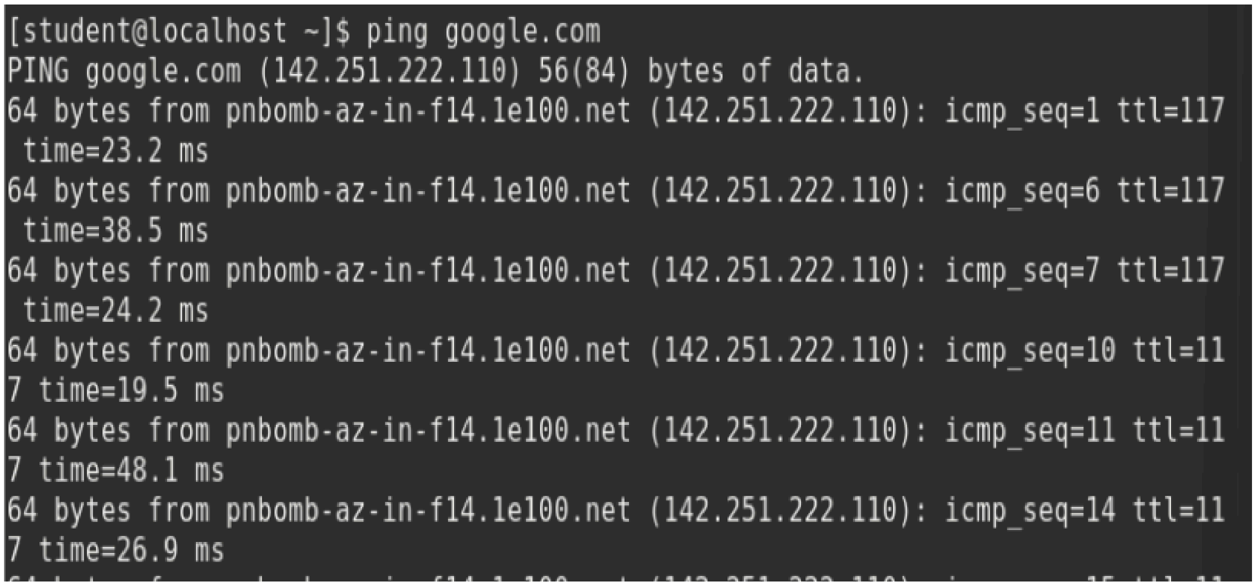
```
[student@localhost ~]$ whois 8.8.8.8
[Querying whois.arin.net]
[whois.arin.net]

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2025, American Registry for Internet Numbers, Ltd.
#


NetRange:       8.8.8.0 - 8.8.8.255
CIDR:           8.8.8.0/24
NetName:        GOGL
NetHandle:      NET-8-8-8-0-2
Parent:         NET8 (NET-8-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Google LLC (GOGL)
RegDate:        2023-12-28
```

**dig** geeksforgeeks.org - dig [server] [name] [type] - domain information groper - It is used for retrieving information about DNS name servers.



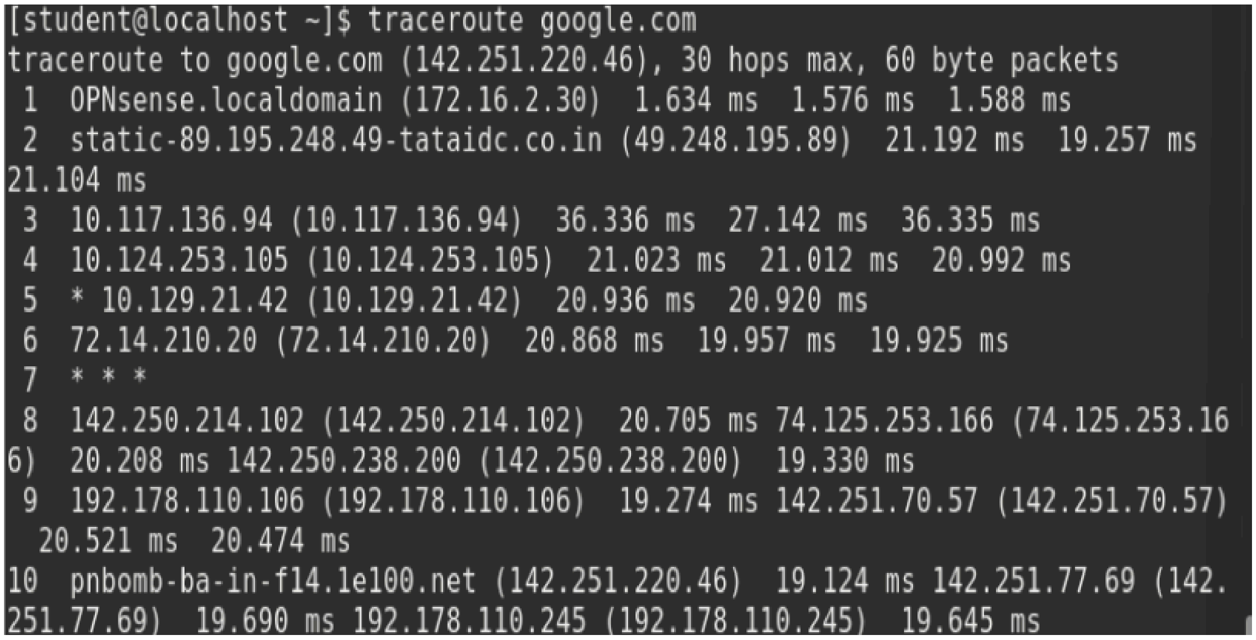
```
[student@localhost ~]$ dig geeksforgeeks.org

; <<>> DiG 9.11.5-P4-RedHat-9.11.5-13.P4.fc30 <<>> geeksforgeeks.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52736
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;geeksforgeeks.org.             IN      A

;; ANSWER SECTION:
geeksforgeeks.org.      63      IN      A       34.218.62.116

;; Query time: 0 msec
;; SERVER: 172.16.2.30#53(172.16.2.30)
;; WHEN: Fri Jul 11 15:17:19 IST 2025
;; MSG SIZE  rcvd: 62
```

**ping** google.com - ping hostname or IP address -  (Packet Internet Groper) - It is used to test connectivity between two nodes.



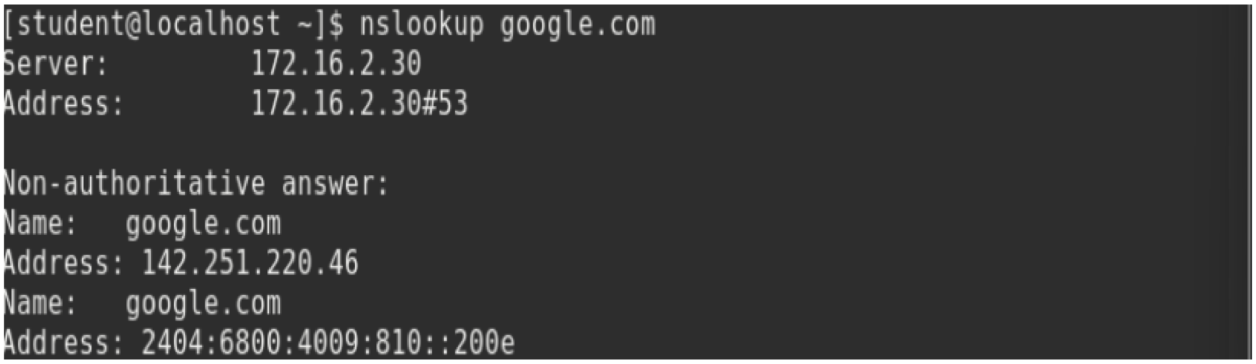
```
[student@localhost ~]$ ping google.com
PING google.com (142.251.222.110) 56(84) bytes of data.
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=1 ttl=117
 time=23.2 ms
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=6 ttl=117
 time=38.5 ms
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=7 ttl=117
 time=24.2 ms
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=10 ttl=11
7 time=19.5 ms
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=11 ttl=11
7 time=48.1 ms
64 bytes from pnbomb-az-in-f14.1e100.net (142.251.222.110): icmp_seq=14 ttl=11
7 time=26.9 ms
```

**traceroute** google.com - traceroute domain name or IP address - tool used to trace the path packets take from a source to a destination.



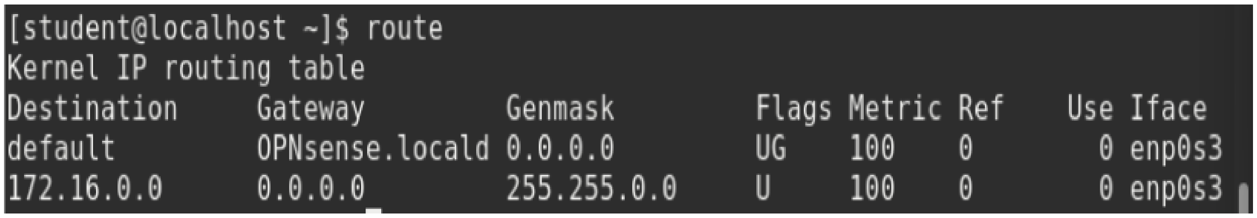
```
[student@localhost ~]$ traceroute google.com
traceroute to google.com (142.251.220.46), 30 hops max, 60 byte packets
 1  OPNsense.localdomain (172.16.2.30)  1.634 ms  1.576 ms  1.588 ms
 2  static-89.195.248.49-tataidc.co.in (49.248.195.89)  21.192 ms  19.257 ms
21.104 ms
 3  10.117.136.94 (10.117.136.94)  36.336 ms  27.142 ms  36.335 ms
 4  10.124.253.105 (10.124.253.105)  21.023 ms  21.012 ms  20.992 ms
 5  * 10.129.21.42 (10.129.21.42)  20.936 ms  20.920 ms
 6  72.14.210.20 (72.14.210.20)  20.868 ms  19.957 ms  19.925 ms
 7  * * *
 8  142.250.214.102 (142.250.214.102)  20.705 ms 74.125.253.166 (74.125.253.16
6)  20.208 ms 142.250.238.200 (142.250.238.200)  19.330 ms
 9  192.178.110.106 (192.178.110.106)  19.274 ms 142.251.70.57 (142.251.70.57)
  20.521 ms  20.474 ms
10  pnbomb-ba-in-f14.1e100.net (142.251.220.46)  19.124 ms 142.251.77.69 (142.
251.77.69)  19.690 ms 192.178.110.245 (192.178.110.245)  19.645 ms
```

**nslookup** google.com - nslookup domain name or IP address - fetch the IP address or the domain name from DNS records.



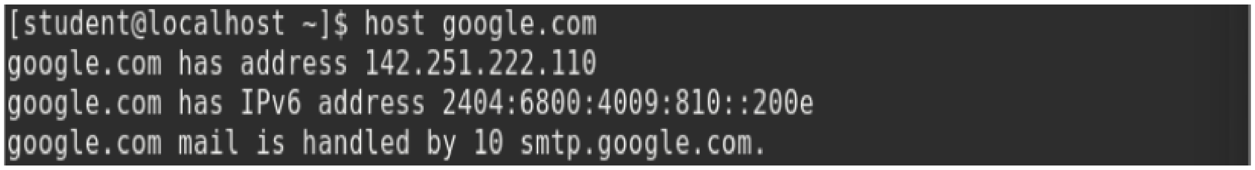
```
[student@localhost ~]$ nslookup google.com
Server:         172.16.2.30
Address:        172.16.2.30#53

Non-authoritative answer:
Name:   google.com
Address: 142.251.220.46
Name:   google.com
Address: 2404:6800:4009:810::200e
```

**route** - shows and manipulates the ip routing table



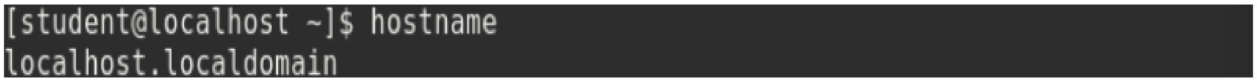
```
[student@localhost ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         OPNsense.locald 0.0.0.0         UG    100    0        0 enp0s3
172.16.0.0      0.0.0.0         255.255.0.0     U     100    0        0 enp0s3
```

**host** - to find a domain name associated with the IP address or find an IP address associated with the domain name.



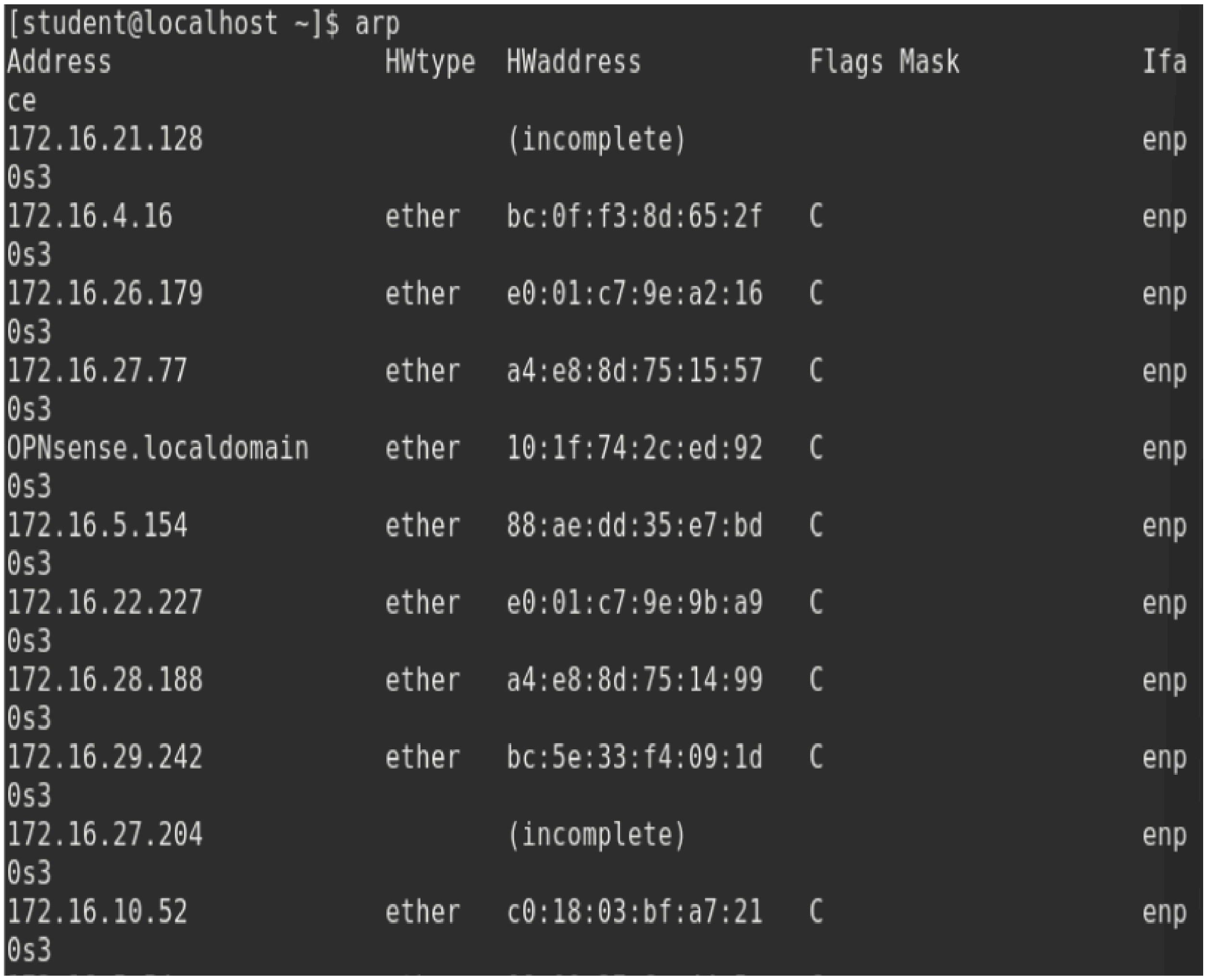
```
[student@localhost ~]$ host google.com
google.com has address 142.251.222.110
google.com has IPv6 address 2404:6800:4009:810::200e
google.com mail is handled by 10 smtp.google.com.
```

**hostname** - To check and set the hostname of the server.



```
[student@localhost ~]$ hostname
localhost.localdomain
```

**arp** - ARP(Address Resolution Protocol) - display and modify ARP cache, which contains the mapping of IP address to MAC address.



```
[student@localhost ~]$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifa
ce
172.16.21.128                    (incomplete)                              enp
0s3
172.16.4.16              ether   bc:0f:f3:8d:65:2f   C                     enp
0s3
172.16.26.179            ether   e0:01:c7:9e:a2:16   C                     enp
0s3
172.16.27.77             ether   a4:e8:8d:75:15:57   C                     enp
0s3
OPNsense.localdomain     ether   10:1f:74:2c:ed:92   C                     enp
0s3
172.16.5.154             ether   88:ae:dd:35:e7:bd   C                     enp
0s3
172.16.22.227            ether   e0:01:c7:9e:9b:a9   C                     enp
0s3
172.16.28.188            ether   a4:e8:8d:75:14:99   C                     enp
0s3
172.16.29.242            ether   bc:5e:33:f4:09:1d   C                     enp
0s3
172.16.27.204                    (incomplete)                              enp
0s3
172.16.10.52             ether   c0:18:03:bf:a7:21   C                     enp
0s3
```

**7.Download and install Nmap. Use it with different options to scan open ports,perform OS, fingerprinting, do a ping scan, tcp port scan, udp port scan**

1. How to Install Nmap
On Linux (Ubuntu/Debian-based)
**sudo apt update**
**sudo apt install nmap -y OR sudo apt-get install nmap**
2. For version detection:
**nmap -sV <target-host's URL or IP>**
3. To scan a System with Hostname and IP address. First, Scan using Hostname
**nmap www.geeksforgeeks.org**
Now let’s Scan using IP Address
**nmap 172.217.27.174**
4. To scan using “-v” option: It is used to get more detailed information about the
remote machines. **nmap -v www.geeksforgeeks.org**
5.To scan multiple hosts: We can scan multiple hosts by writing IP addresses or
hostnames with **nmap. nmap 103.76.228.244 157.240.198.35 172.217.27.174**
6.To scan whole subnet: We can scan a whole subnet or IP range with nmap by
providing “*” with it. It will scan a whole subnet and give the information about
those hosts which are Up in the Network. **nmap 103.76.228.***
7.To scan specific range of IP address: We can specify the range of IP addresses. This command will scan IP address 192.168.29.1 to 192.168.29.20 . **nmap 192.168.29.1-20**

8.To scan to detect firewall settings: Detecting firewall settings can be useful
during penetration testing and vulnerability scans. To detect it we use “-sA” option. This will provide you with information about firewall being active on the host. It
uses an ACK scan to receive the information. **sudo nmap -sA 103.76.228.244**
9.Here -sS flag is used for TCP SYN Scan, Which is a stealthy and efficient
method of scanning for open ports on a target system. **sudo nmap -sS <Domain Name>**
10. The “-sn” flag is used with nmap to perform a ping scan, which sends ICMP requests to a target host or network to determine hosts is up or not. This checks which devices are alive in the network without scanning ports.
**nmap -sn <Domain Name>**
**sudo nmap -sU <domain Name> ----> (UDP)**
11.The “-p” flag is used with nmap to perform scan on a specific port or range of ports. ( In our case it will scan port 80,443 and 21 )

**nmap -p 80 443 21 <Domain Name>**
12.We can also specify the range of ports to scan on a network. ( In this case it
will scan all the ports in the range of 1 to 80 )
**nmap -p 1-80 <Domain Name>**
13.Here -A indicates aggressive, it will give us extra information, like OS
detection (-O), version detection, script scanning (-sC), and traceroute (-
traceroute). It even provides a lot of valuable information about the host. **nmap -A <Domain Name>**
14.Using this command we can discover the target hosting service or identify
additional targets according to our needs for quickly tracing the path.
**nmap --trace out <Domain Name>**
15. Here it will display the operating system **sudo nmap -O <Domain Name>**

**8.Implement a code to simulate DOS attack.**

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char buff[15];// buffer size = 15 bytes
    int pass = 0;

    printf("\n Enter the password: \n");
    gets(buff); // ?? UNSAFE: no bounds checking

    if(strcmp(buff, "thecorrectpaswd")) {
        printf("\n Wrong Password \n");
    }
    else {
        printf("\n Correct Password \n");
        pass = 1;
    }

    if(pass) {
    printf("\n Root privileges given to the user \n");
    }

    return 0;
}
```

```
gcc p8.c -o p8

p8.exe

Enter the password:
thecorrectpaswd
 Correct Password
 Root privileges given to the user
p8.exe
 Enter the password:
vbjfdbvk
 Wrong Password
```

9.Write a Python program to implement the Playfair cipher using Substitution technique.

```
# Exp9: Implement Python program to illustrate Playfair Cipher technique

# Step 1: Create the 5x5 key matrix
def generate_key_matrix(key):
    key = key.lower().replace('j', 'i')  # Replace 'j' with 'i'
    matrix = []
    used = set()

    for char in key:
        if char.isalpha() and char not in used:
            used.add(char)
            matrix.append(char)

    # Fill remaining letters (excluding 'j')
    for char in 'abcdefghiklmnopqrstuvwxyz':
        if char not in used:
            used.add(char)
            matrix.append(char)

    # Convert list into 5x5 matrix
    return [matrix[i*5:(i+1)*5] for i in range(5)]

# Step 2: Preprocess plaintext into digraphs
def preprocess_text(text):
    text = text.lower().replace('j', 'i').replace(' ', '')
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = 'x'
        if i + 1 < len(text):
            b = text[i + 1]
            if a == b:
                pairs.append((a, 'x'))
                i += 1
            else:
                pairs.append((a, b))
                i += 2
        else:
            pairs.append((a, 'x'))
            i += 1
    return pairs

# Step 3: Find character position in matrix
def find_position(matrix, char):
    for row in range(5):
        for col in range(5):
            if matrix[row][col] == char:
                return row, col
    return None

# Step 4: Encrypt each pair using Playfair rules
def encrypt_pair(pair, matrix):
    a, b = pair
    row1, col1 = find_position(matrix, a)
    row2, col2 = find_position(matrix, b)

    if row1 == row2:
        # Same row → shift right
        return matrix[row1][(col1 + 1) % 5] + matrix[row2][(col2 + 1) % 5]
    elif col1 == col2:
        # Same column → shift down
        return matrix[(row1 + 1) % 5][col1] + matrix[(row2 + 1) % 5][col2]
    else:
        # Rectangle rule → swap columns
        return matrix[row1][col2] + matrix[row2][col1]

# Step 5: Full encryption function
def playfair_encrypt(plaintext, key):
    matrix = generate_key_matrix(key)
    print("Key Matrix:")
    for row in matrix:
        print(row)

    pairs = preprocess_text(plaintext)
    print("\nDigraphs:", pairs)

    ciphertext = ''
    for pair in pairs:
        encrypted = encrypt_pair(pair, matrix)
        print(f"{pair[0]}{pair[1]} → {encrypted}")
        ciphertext += encrypted

    return ciphertext.upper()

# Main program
plaintext = input("Enter text to encrypt: ")
key = input("Enter key: ")
ciphertext = playfair_encrypt(plaintext, key)
print("\nCiphertext:", ciphertext)
```

```
Enter text to encrypt: Hello world
Enter key: monarchy
Key Matrix:
['m', 'o', 'n', 'a', 'r']
['c', 'h', 'y', 'b', 'd']
['e', 'f', 'g', 'i', 'k']
['l', 'p', 'q', 's', 't']
['u', 'v', 'w', 'x', 'z']

Digraphs: [('h', 'e'), ('l', 'x'), ('l', 'o'), ('w', 'o'), ('r', 'l'),
('d', 'x')]
he → cf
lx → su
lo → pm
wo → vn
rl → mt
dx → bz

Ciphertext: CFSUPMVNMTBZ
```